Introduction to

DIGITAL COMMUNICATION RECEIVER DESIGN

Prepared by

C. RICHARD JOHNSON JR.

for delivery at University College Dublin (Ireland) and Technische Universiteit Delft (the Netherlands) in

APRIL-MAY 2006

under the support of a Fulbright Scholarship and a Weiss Fellowship.

- Lectures drawn from Johnson and Sethares, *Telecommunication Breakdown: Concepts of Communication Transmitted via Software-Defined Radio* (Prentice Hall, 2004).
- Lab assignments use a Matlab-based PAM Radio from Dr. Andy Klein.
- Distribution does not constitute release of copyright. All rights reserved.

Five Day Schedule

- **DAY 1** (3 lecture hours, 3 lab hours)
  - A (Naive) Digital Radio
  - (De)Modulation
  - Automatic Gain Control
- **DAY 2** (3 lecture hours, 3 lab hours)
  - An Idealized RF System Simulation
  - Carrier Recovery
- **DAY 3** (3 lecture hours, 3 lab hours)
  - Pulse Shaping and Receive Filtering
  - Baud Timing for Clock Recovery
- **DAY 4** (3 lecture hours, 3 lab hours)
  - Linear Equalization
  - Putting It All Together
- **DAY 5** (6 lab hours)
  - Design Project and Report Preparation
  - Design Testing and Report Presentation

This compacted 5-day introduction to digital communication recevier design was originally extracted from C. R. Johnson, Jr. and W. A. Sethares, *Telecommunication Breakdown: Concepts of Communication Transmitted via Software-Defined Radio* (Prentice Hall, 2004) under the support of Prof. Rick Johnson by a Fulbright Scholarship to France in the latter half of 2005. The accompanying labs were developed in collaboration with Dr. A. G. Klein (currently a post-doctoral researcher in the Laboratoire de Signaux et Systemes, Supélec, Gif sur Yvette, France). The first version of this compacted course was offered in the ATHENS Programme at École Nationale Supérieure des Télécommunications (Paris, France) in November 2005. The current version was prepared for presentation in April and May 2006 at University College Dublin (Ireland) and Technische Universiteit Delft (the Netherlands). This spring 2006 teaching activity is supported in part by a Stephen H. Weiss Preisdential Fellowship from Cornell University.

In keeping with the philosophy of *Telecommunication Breakdown*, this compacted version is built around a Matlab-based software radio (developed by Dr. Andy Klein) that implements the major digital signal processing operations of a common radio receiver: demodulation, carrier recovery, matched receive filtering, baud-timing, equalization, and decoding. This radio can compensate for the transmission impairments of carrier phase jitter, channel noise, time-varying channel intersymbol interference, and baud-timing offset. Relying on a background in signals and systems comparable to that of J. H. McClellan, R. W. Schafer, and M. A. Yoder, *Signal Processing First* (Pearson Prentice Hall, 2003), DAY 1 lectures present a basic pulse-amplitude-modulated (PAM) radio system and discuss how such impairments, if uncompensated, can deteriorate communication system performance. A basic adaptive algorithm creation strategy is described (in particular for automatic gain control) to track the compensator parameter needed to counteract an encountered impairment, the specifics of which are initially unknown to the user and expected to be time-varying. DAY 2 lectures feature simulated system performance degradation due to various impairments, and the successful automatic gain control compensation of flat fading. The adaptive (stochastic gradient descent based) strategy is applied on DAY 2 to carrier phase tracking by the receiver mixer (resulting in the popular phase-locked and Costas loop algorithms), on DAY 3 to baud timing for clock recovery (based on downsampled signal power optimization), and on DAY 4 to equalization of frequency-selective channel impairments (via both trained and blind schemes). DAY 5 consists of a final project assignment that puts it all together.

The first 4 days are designed for 3 hours of lecture followed by 3 hours of supervised lab instruction. There are no lectures on the 5th day, just a lab session, by the end of which the modified radio developed individually by each student will be tested in comparison to the base radio developed by Dr. Klein.

This course packet provides the overheads used in the lectures of days 1-4, the associated lab assignments for days 1-4, the description of the final project, and a user's manual for Dr. Klein's software radio. An accompanying CD includes the pertinent Matlab files (designed for compatibility with version 6) for demonstrations cited in lecture, Dr. Klein's software radio, the labs, and final project. The CD also includes a pdf version of the printed course packet. As a bonus, a movie is also included of a working receiver built (in the late 1980s) by Applied Signal Technology for 16-QAM (where carrier phase offset results in rotation of the recovered 4 by 4 constellation and carrier frequency offset results in recovered constellation

rotation). A document, which is drawn from the CD accompanying *Telecommunication Breakdown* and also includes a Matlab simulated radio, on the extension of the PAM radio of *Telecommunication Breakdown* – and this compacted course – to the more pragmatic QAM radio is also included on the CD for this compacted course, along with all of the Matlab-based software for its simulation.

DAY 1

- A (Naive) Digital Radio
- (De)Modulation
- Automatic Gain Control

A (NAIVE) DIGITAL RADIO

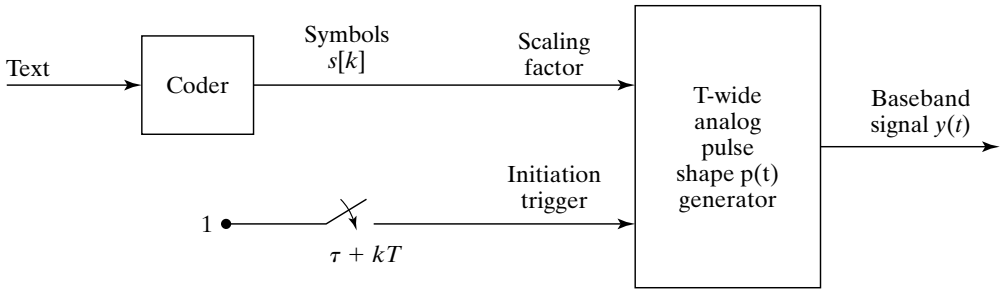
- ⋆ An Illustrative Digital Communication System
- ⋆ Transmitter and Transmitted Pulse Sequence
- ⋆ Received Signal and Receiver
- ⋆ Synchronization Issues
- ⋆ Spectrum Sharing
- ⋆ RF Communication System
- ⋆ Practical Obstacles
- ⋆ Analog/Digital Signal Processing Split

An Illustrative Digital Communication System

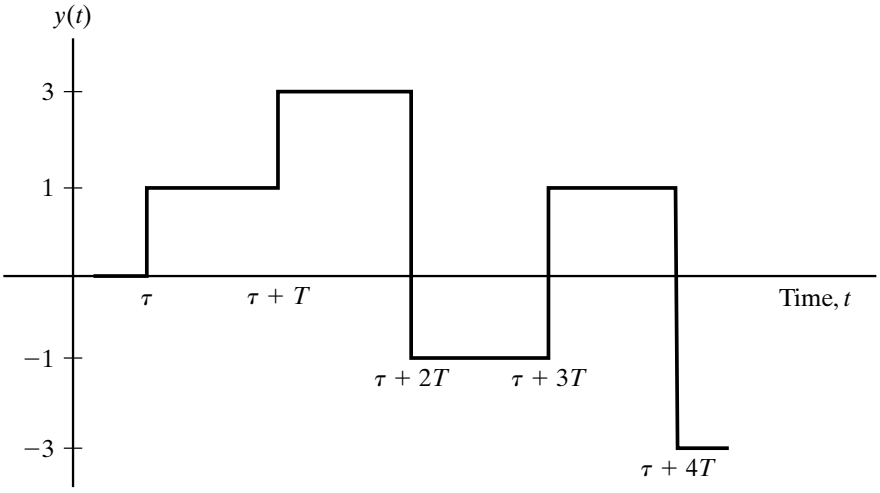
- Objective: Send text converted to a stream of bits from place 1 to place 2 through the analog medium in between.
- Coding: Use standard ASCII code to convert text to bits (using 8 bits per character)
- Transmitter: Use sequence of scaled rectangular pulses to convey bits singly, e.g., $1 \rightarrow +1$ and $0 \rightarrow -1$ or in clusters, e.g., $10 \rightarrow +1$, $01 \rightarrow -1$, $00 \rightarrow +3$, and $11 \rightarrow -3$. We choose pairs, so groups of 8 bits become clumps of 4 symbols.
- Receiver: Sample received pulse and convert symbols to bits, e.g., $1, 3, -1, 1, -3 \rightarrow 1000011011$, and then back to text.

Transmitter and Transmitted Pulse Sequence

- An idealized baseband transmitter



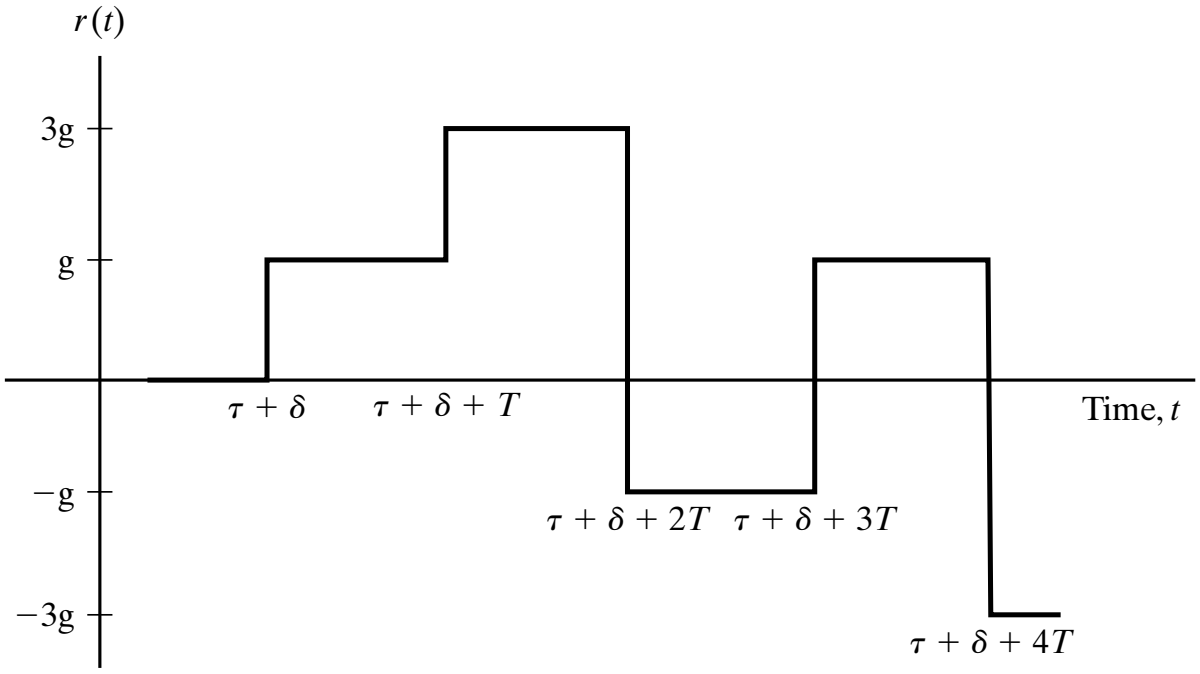
  and transmitted (baseband) signal



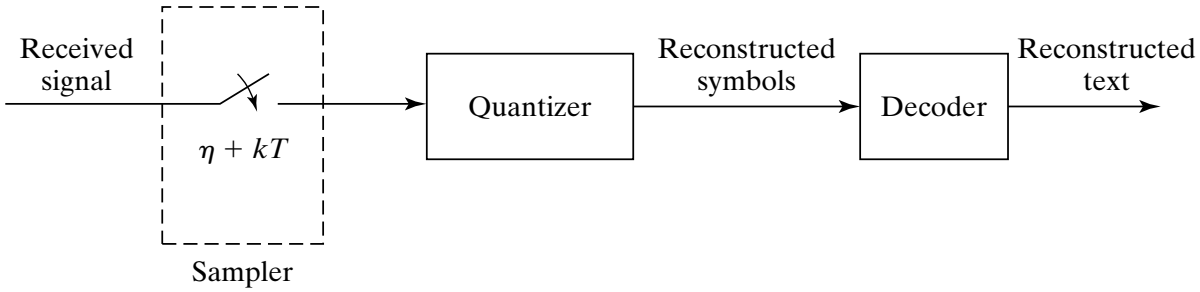
- The transmitted signal consists of a sequence of pulses, one corresponding to each symbol.
- Each pulse has the same rectangular shape though offset in time and scaled in magnitude.

Received Signal and Receiver

- In the ideal case, the received signal is the same as the transmitted signal though attenuated in magnitude and delayed in time.



- An idealized baseband receiver



Synchronization Issues

- Baud (symbol) timing
  - ⊙ η selection for fixed T
  - ⊙ top-dead-center

$$\eta = \tau + \delta + T/2$$

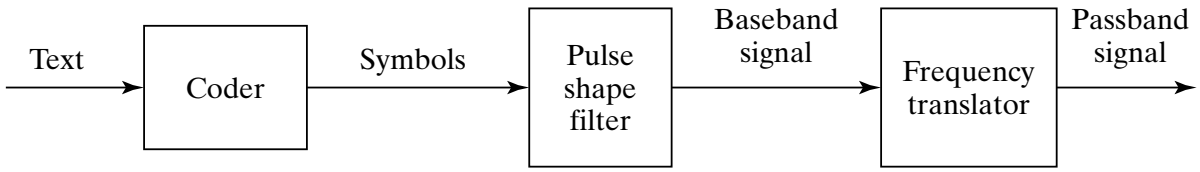
  - ⊙ Peaked (rather than rectangular) pulse shapes will reduce the spectral footprint of the sequence of pulses, but increase the sensitivity to top-dead-center baud-timing.
- Frame start determination
  - ◦ grouping symbols to decoder
  - ◦ example: $-1, -1, 1, -3, -1$; first 4 symbols decode to "X" and last four decode to "a"
  - ◦ special marker sequence inserted in source sequence at start of a frame with subsequent frame starts determined by knowledge of the the period of their recurrence.

Spectrum Sharing

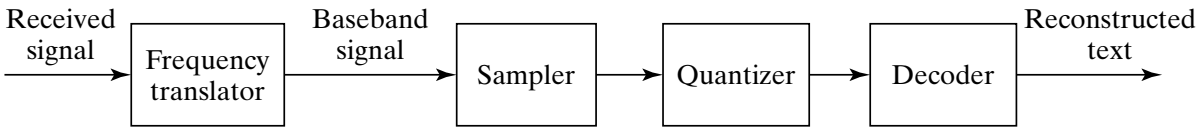
- Several user pairs should be able to communicate through same medium simultaneously in same geographical region.
- Interference avoidance achieved by disallowing use of same frequencies by different users in same geographical area.
- Bandwidth occupied by pulse shape/sequence is inversely related to rectangle width.
- More frequent symbol transmission achieved by narrower pulses increases exclusionary baseband spectrum requirement.
- If all frequencies in bandlimited baseband spectrum can be translated by same amount, several users could be multiplexed to different center frequencies without overlap.

Radio Frequency (RF) Communication System

- RF transmitter

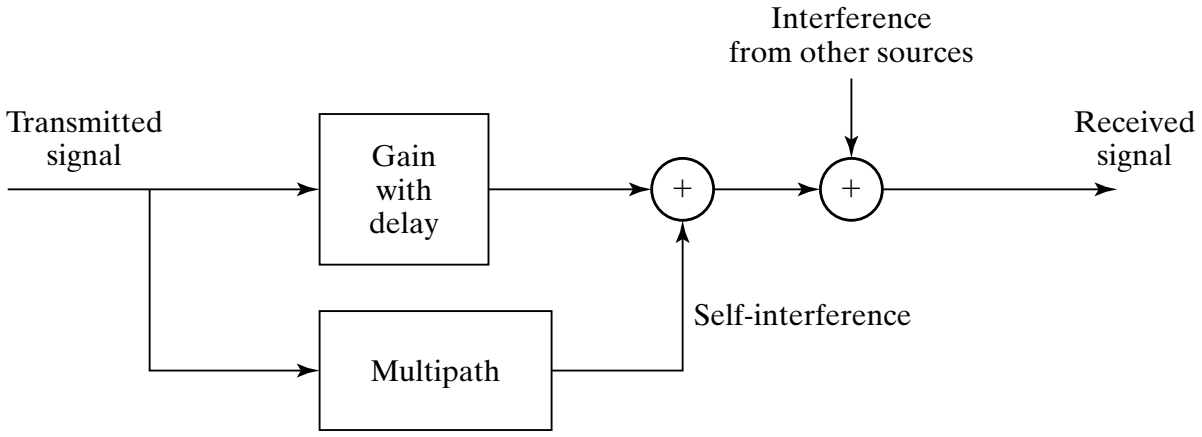


- RF receiver



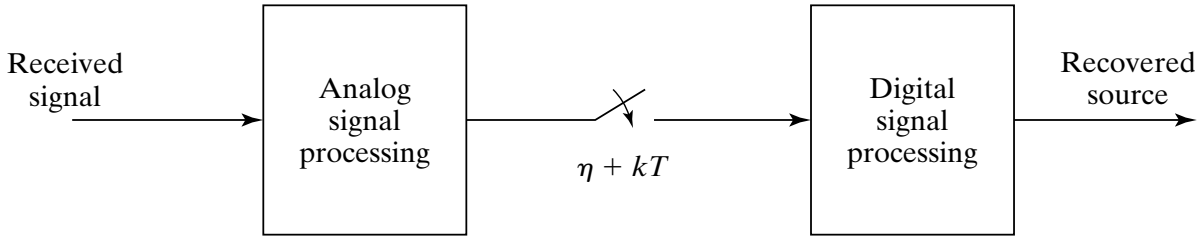
Practical Obstacles

- precise frequency translation required in receiver
- precise timing required in receiver
- multi-user interference occurs in received signal, e.g. since each user is not strictly bandlimited in frequency
- noise contamination of transmitted signal: in-band, out-of-band, narrowband, or broadband
- channel distortion: fading or multipath, possibly time-varying



Analog/Digital Signal Processing Split

- Due to cost and flexibility benefits, modern radio design is pushing the sampler (and subsequent digital signal processing) closer to the received signal, i.e. the output of the low noise amplifier driven by the antenna signal.



- Sample period $\leq$ symbol period.

An ASP/DSP Division of Labor

ASP:

- frequency translation to intermediate frequency
- out-of-band signal attenuation
- automatic gain control

DSP:

- downconversion to baseband (via mixer)
- carrier tracking (via mixer phase setting)
- symbol timing (via interpolation)
- channel compensation (via linear filtering)
- symbol decision (via quantization)
- frame synchronization (via marker correlation)
- decode symbols to message text (via table)

(DE)MODULATION

- ⋆ Up-Conversion via Mixing
- ⋆ Downconversion via Mixing
- ⋆ Message Recovery via Filtering
- ⋆ Synchronized Demodulation of Amplitude Modulation with Suppressed Carrier
- ⋆ Unsynchronized Demodulation
- ⋆ Sub-Nyquist Sampling of RF
- ⋆ Interpolation

Up-Conversion via Mixing

- For upconversion mixer multiplies input waveform with a sinusoid
  - $s(t) = w(t)\cos(2\pi f_o t)$
  - $w(t)$: message waveform
  - $s(t)$: transmitted waveform (mixer output)
- We want to compute the Fourier transform of the transmitted waveform $s(t)$ using:
  - Exponential definition of a cosine

$$\cos(x) = \frac{1}{2}(e^{jx} + e^{-jx})$$

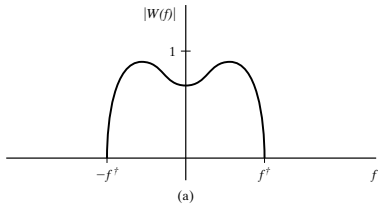
  - Fourier transform definition

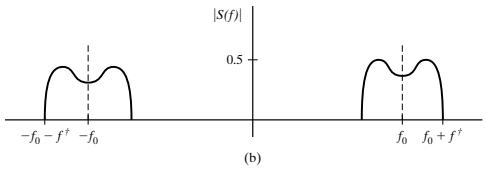
$$W(f) = \int_{-\infty}^{\infty} w(t)e^{-j2\pi ft}dt = \mathcal{F}\{w(t)\}$$

Up-Conversion via Mixing (cont'd)

- So:

$$S(f) = \mathcal{F}\{s(t)\} = \mathcal{F}\{w(t)\ \cos(2\pi f_0 t)\}$$
$$= \mathcal{F}\left\{w(t)\left[\tfrac{1}{2}\left(e^{j2\pi f_0 t} + e^{-j2\pi f_0 t}\right)\right]\right\}$$
$$=$$
$$\int_{-\infty}^{\infty} w(t)\left[\tfrac{1}{2}\left(e^{j2\pi f_0 t} + e^{-j2\pi f_0 t}\right)\right] e^{-j2\pi f t} dt$$
$$=$$
$$\tfrac{1}{2}\int_{-\infty}^{\infty} w(t)\left(e^{-j2\pi(f-f_0)t} + e^{-j2\pi(f+f_0)t}\right) dt$$
$$= \tfrac{1}{2}\int_{-\infty}^{\infty} w(t)\ e^{-j2\pi(f-f_0)t} dt$$
$$+\tfrac{1}{2}\int_{-\infty}^{\infty} w(t)\ e^{-j2\pi(f+f_0)t} dt$$
$$= \tfrac{1}{2}W(f - f_0) + \tfrac{1}{2}W(f + f_0)$$





Downconversion via Mixing

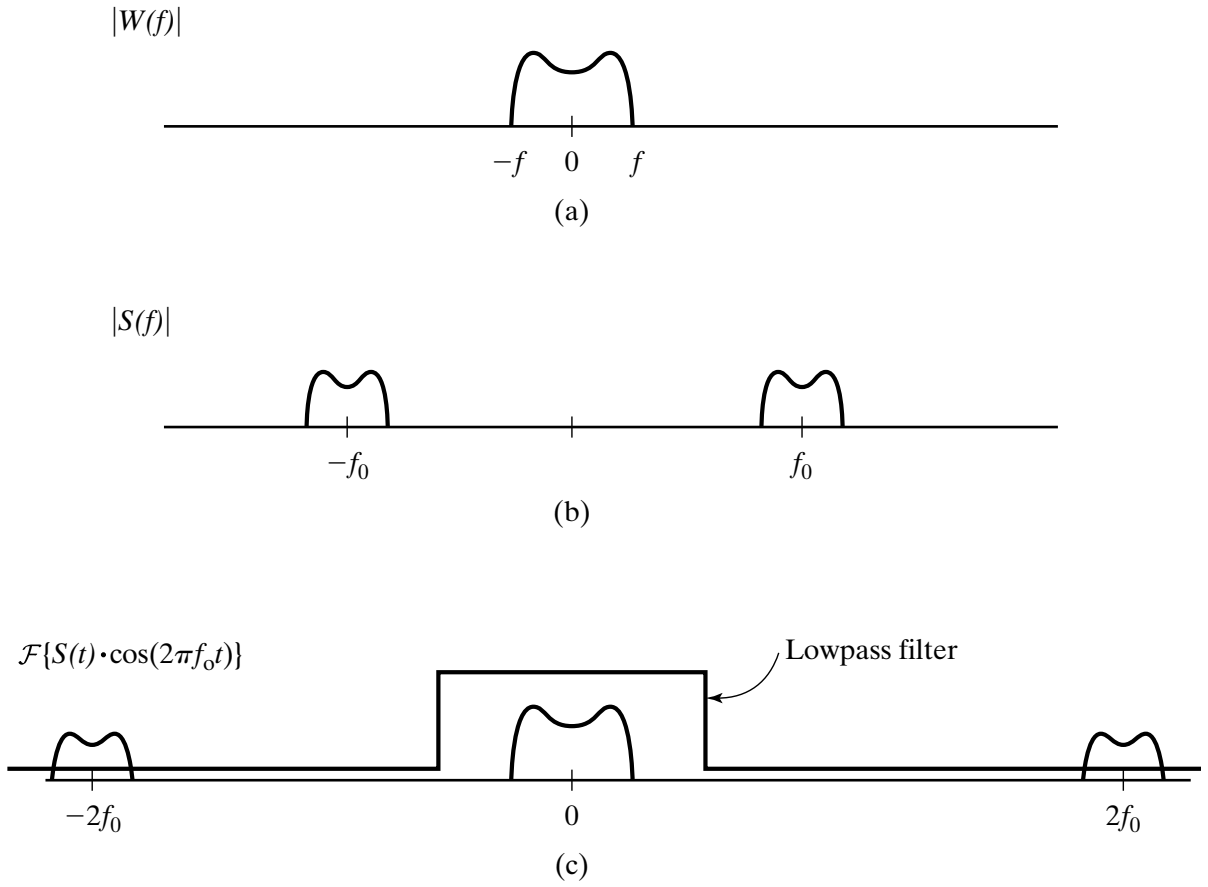
- Assume transmitted signal arrives unimpaired
- For downconversion use mixer with frequency and phase matching transmitter's
  - $d(t) = s(t)\cos(2\pi f_0 t) = w(t)\cos^2(2\pi f_0 t)$
  - $\cos^2(x) = \frac{1}{2} + \frac{1}{2}\cos(2x)$
  - $d(t) = w(t)\left[\frac{1}{2} + \frac{1}{2}\cos(4\pi f_0 t)\right]$
    $= \frac{1}{2}w(t) + \frac{1}{2}w(t)\cos(2\pi(2f_0)t)$
- Using linearity of Fourier transform and previously extracted result on Fourier transform of mixer output

$$\begin{aligned} D(f) &= \mathcal{F}\{d(t)\} \\ &= \mathcal{F}\{\tfrac{1}{2}w(t) + \tfrac{1}{2}w(t)\cos(2\pi(2f_0)t)\} \\ &= \tfrac{1}{2}\mathcal{F}\{w(t)\} + \tfrac{1}{2}\mathcal{F}\{w(t)\cos(2\pi(2f_0)t)\} \\ &= \tfrac{1}{2}W(f) + \tfrac{1}{4}W(f-2f_0) + \tfrac{1}{4}W(f+2f_0) \end{aligned}$$

Message Recovery via Filtering

- Passing a signal $s(t)$ through a linear system with transfer function $h(t)$ results in an output that is the convolution of $s(t)$ and $h(t)$.
- The Fourier transform of a convolution is the product of the Fourier transforms.
- We often distinguish among linear systems based on the range of frequencies they pass or reject, e.g. lowpass, highpass, bandpass, notch.
- The $\frac{1}{2}W(f)$ portion of $D(f)$ about zero frequency can be extracted by filtering $d(t)$ through an ideal filter that has a flat magnitude (and a linear phase) for low frequencies and (near) zero magnitude for high frequencies, i.e. an ideal lowpass filter.

Message Recovery via Filtering (cont'd)



(a) original spectrum of the message

(b) message modulated by the carrier

(c) demodulated signal has original spectrum after ideal lowpass filtering

Synchronized Demodulation of Amplitude Modulation with Suppressed Carrier

- *analog message signal*: $w(t)$
- *transmitted/modulated signal*:

$$v(t) = A_c w(t) \cos(2\pi f_c t)$$

- *transmitted signal spectrum*:

$$V(f) = \frac{1}{2} A_c W(f + f_c) + \frac{1}{2} A_c W(f - f_c)$$

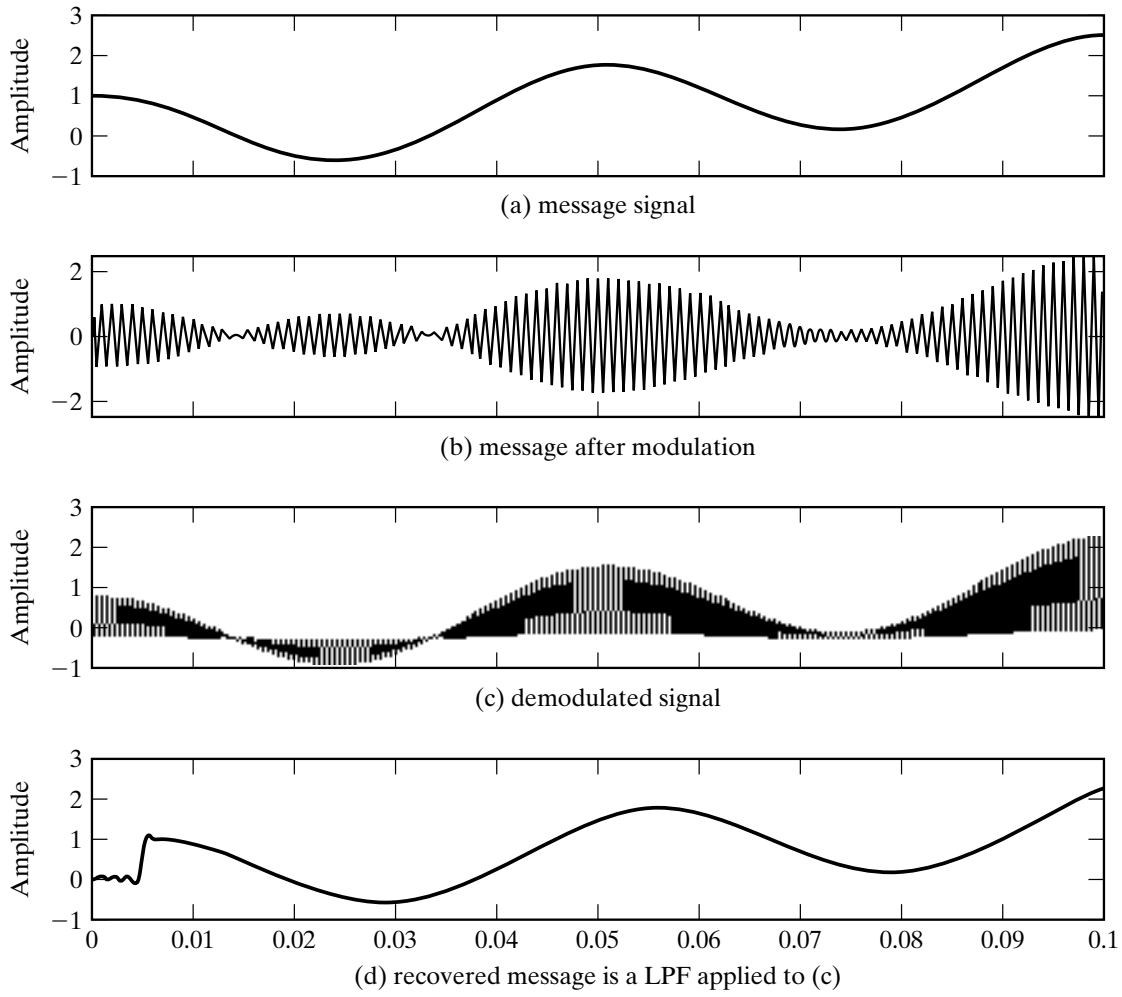
- *ideal demodulation with synchronized mixing and LPF*:

$$m(t) = \text{LPF}\{v(t)\cos(2\pi f_c t)\} = \frac{1}{2} A_c W(f)$$

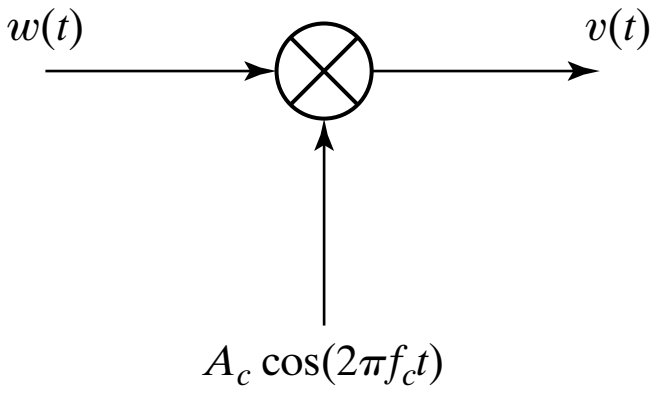
- *main disadvantage*: carrier phase and frequency synchronization needed at receiver

Synchronized Demodulation of Amplitude Modulation with Suppressed Carrier (cont'd)

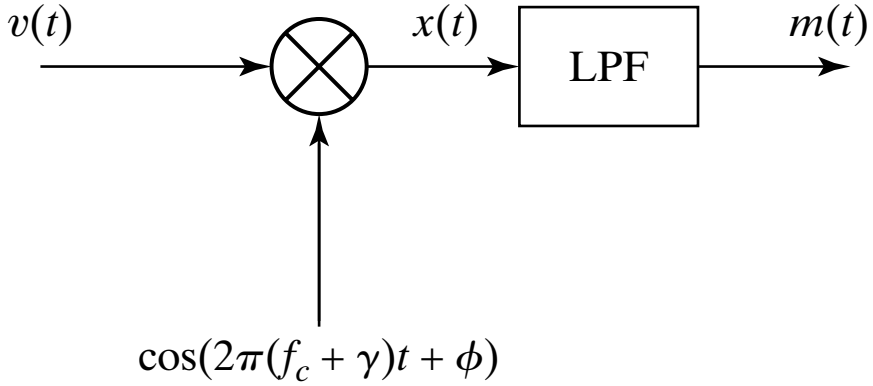
- *Example*: Perfect (delayed) recovery with perfect synchronization using `AM`



Unsynchronized Demodulation



(a)



(b)

(a) transmitter/modulator; (b) unsynchronized receiver/demodulator

Unsynchronized Demodulation (cont'd)

- Using

$$\mathcal{F}\{g(t)\cos(2\pi\alpha t+\theta)\}$$

$$=\frac{1}{2}\left[e^{j\theta}G(f-\alpha)+e^{-j\theta}G(f+\alpha)\right]$$

on

$$x(t)=v(t)\cos(2\pi(f_c+\gamma)t+\phi)$$

and

$$V(f)=\frac{1}{2}A_cW(f+f_c)+\frac{1}{2}A_cW(f-f_c)$$

yields

$$X(f)=\frac{A_c}{4}\left[e^{j\phi}\{W(f+f_c-(f_c+\gamma))\right.$$

$$+W(f-f_c-(f_c+\gamma))\}$$

$$+e^{-j\phi}\{W(f+f_c+(f_c+\gamma))$$

$$\left.+W(f-f_c+(f_c+\gamma))\}\right]$$

$$=\frac{A_c}{4}\left[e^{j\phi}W(f-\gamma)+e^{j\phi}W(f-2f_c-\gamma)\right.$$

$$\left.+e^{-j\phi}W(f+2f_c+\gamma)+e^{-j\phi}W(f+\gamma)\right]$$

Unsynchronized Demodulation (cont'd)

- If no frequency offset ($\gamma = 0$), then with exponential description of cosine

$$X(f) = \frac{A_c}{4}\left[(e^{j\phi} + e^{-j\phi})W(f)\right.$$

$$\left.+e^{j\phi}W(f-2f_c) + e^{-j\phi}W(f+2f_c)\right]$$

$$= \frac{A_c}{2}W(f)\cos(\phi)$$

$$+\frac{A_c}{4}\left[e^{j\phi}W(f-2f_c) + e^{-j\phi}W(f+2f_c)\right]$$

Thus, with LPF cutoff between $W(f)$ bandwidth B and $2f_c - B$

$$m(t) = \text{LPF}\{x(t)\} = \frac{A_c}{2}w(t)\cos(\phi)$$

Recovered signal is attenuated relative to perfectly synchronized demodulation.

As ϕ approaches $\pi/2$, recovered signal vanishes.

Unsynchronized Demodulation (cont'd)

- If no carrier offset ($\phi = 0$),

$$X(f) = \frac{A_c}{4}\left[W(f-\gamma) + W(f-2f_c-\gamma)\right.$$

$$\left. + W(f+2f_c+\gamma) + W(f+\gamma)\right]$$

  Thus, with $m(t) = \text{LPF}\{x(t)\}$

$$M(f) = \frac{A_c}{4}\left[W(f-\gamma) + W(f+\gamma)\right]$$

  and using frequency shifting property of multiplication by a cosine

$$m(t) = \frac{A_c}{2} w(t)\cos(2\pi\gamma t)$$

  Recovered signal is low-frequency amplitude modulated relative to perfectly synchronized demodulation; periodically (every $1/\gamma$ sec) it vanishes.

- Ergo: The need for carrier recovery

Sub-Nyquist Sampling of RF Signal

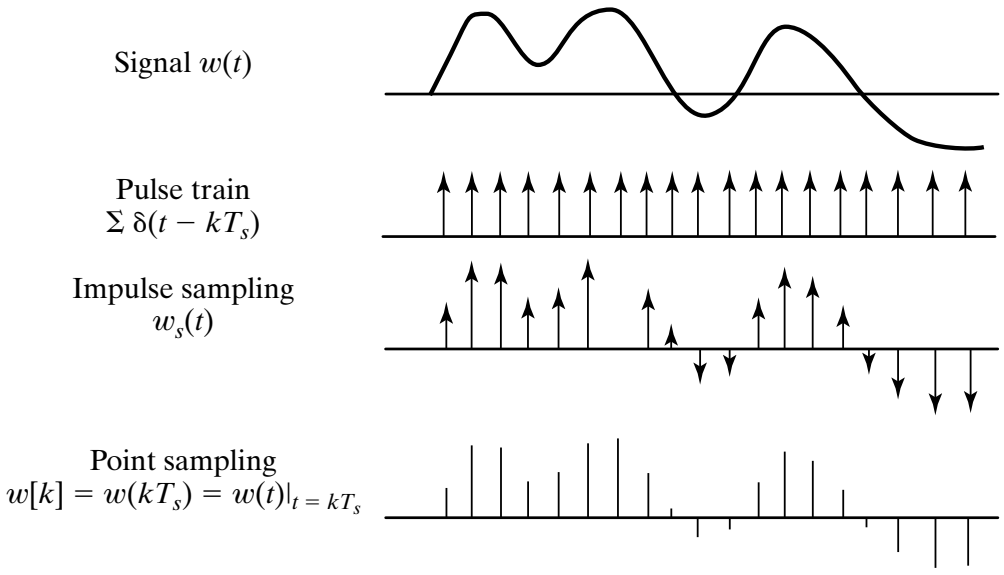
- In a digital radio, the sampler can be after analog demodulation to baseband or after partial analog demodulation to an intermediate frequency.
- With sampling after analog demodulation to baseband, we can use the Nyquist sampling theorem to select a sample rate that allows perfect reconstruction of analog signal at any point in time just from sampled values.
- If we sample before demodulation to baseband, must we sample at (the much higher) Nyquist rate for the RF signal to achieve successful demodulation?

Sub-Nyquist Sampling (cont'd)

With $w(t)$ the input to an impulse sampler, the output $w_s(t)$ is

$$w_s(t) = w(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_s)$$

Analog $w(t)$ is multiplied point-by-point by a pulse train



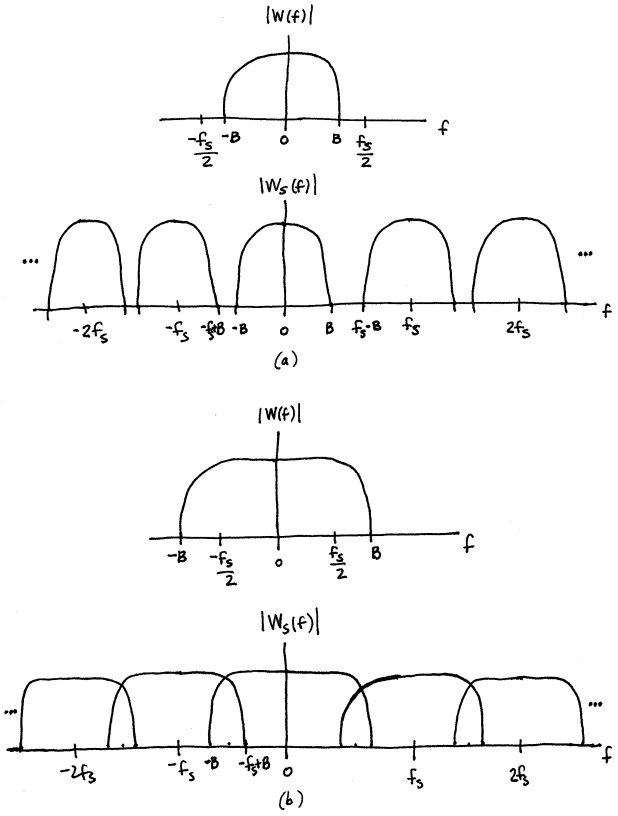
Sub-Nyquist Sampling (cont'd)

- With $f_s = 1/T_s$

$$W_s(f) = f_s \sum_{n=-\infty}^{\infty} W(f - nf_s)$$

  Relative to $W(f)$, $W_s(f)$ has been scaled by f_s and contains replicas at every f_s.

- Largest frequency in $W(f)$ less than $f_s/2$ (top plot) and slightly larger than $f_2/2$ (bottom)



Sub-Nyquist Sampling (cont'd)

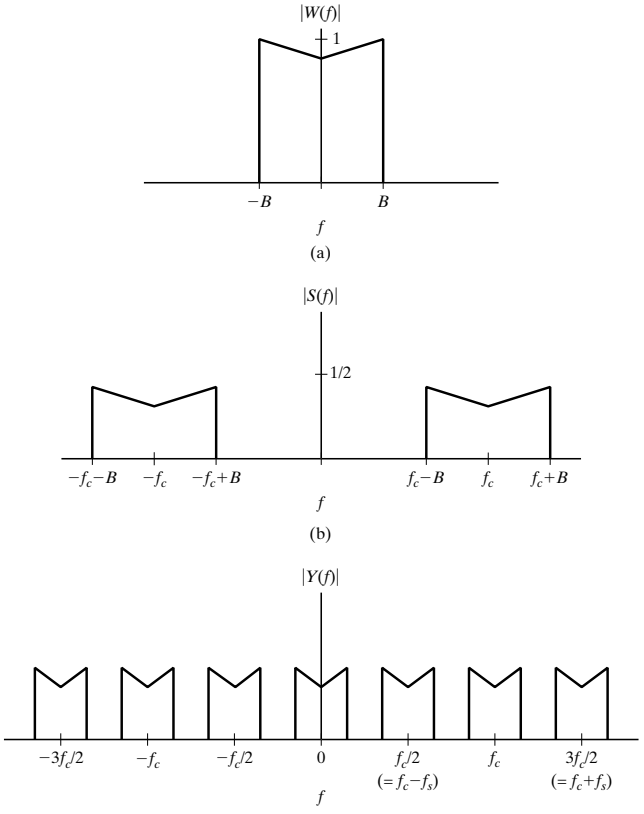
- *Nyquist Sampling Theorem*:

  If the signal $w(t)$ is bandlimited to B, ($W(f) = 0$ for all $|f| > B$) and if the sampling rate is faster than $f_s = 2B$, then $w(t)$ can be reconstructed exactly for all t from its samples $w(kT_s)$.

- *Sub-Nyquist Sampling*:
  - What if the signal to be sampled is a passband signal, but the signal to be reconstructed is this passband signal downconverted to a baseband signal with a much lower maximum frequency?
  - Can sub-Nyquist sampling of the passband signal be employed without aliasing of the baseband signal?
  - The following examples provide a positive answer.

Sub-Nyquist Sampling (cont'd)

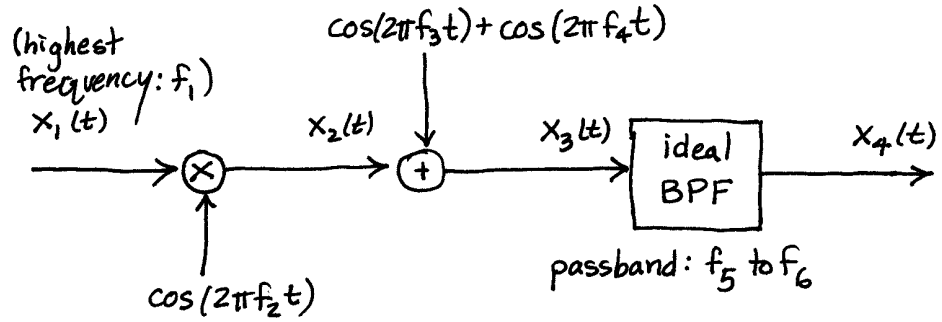
- *Example*:
  - Consider $f_s = f_c/2$

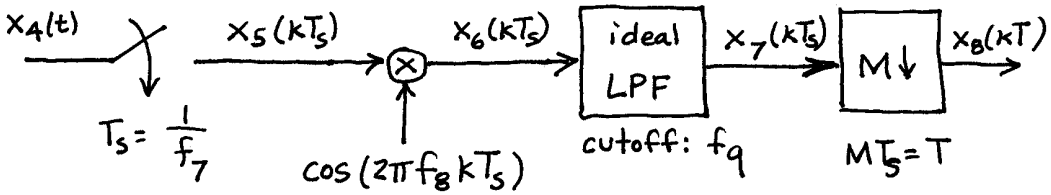


  - Works for $f_s = f_c/n$
  - What if f_s not exactly f_c/n?

Sub-Nyquist Sampling (cont'd)

- *Another Example*: For a PAM system the sampler, downconverter, and downsampler (to symbol period T) should produce an output x_8 with a spectrum matching that of a sampled version (with sample period matching symbol period) of the baseband source x_1.





Sub-Nyquist Sampling (cont'd)

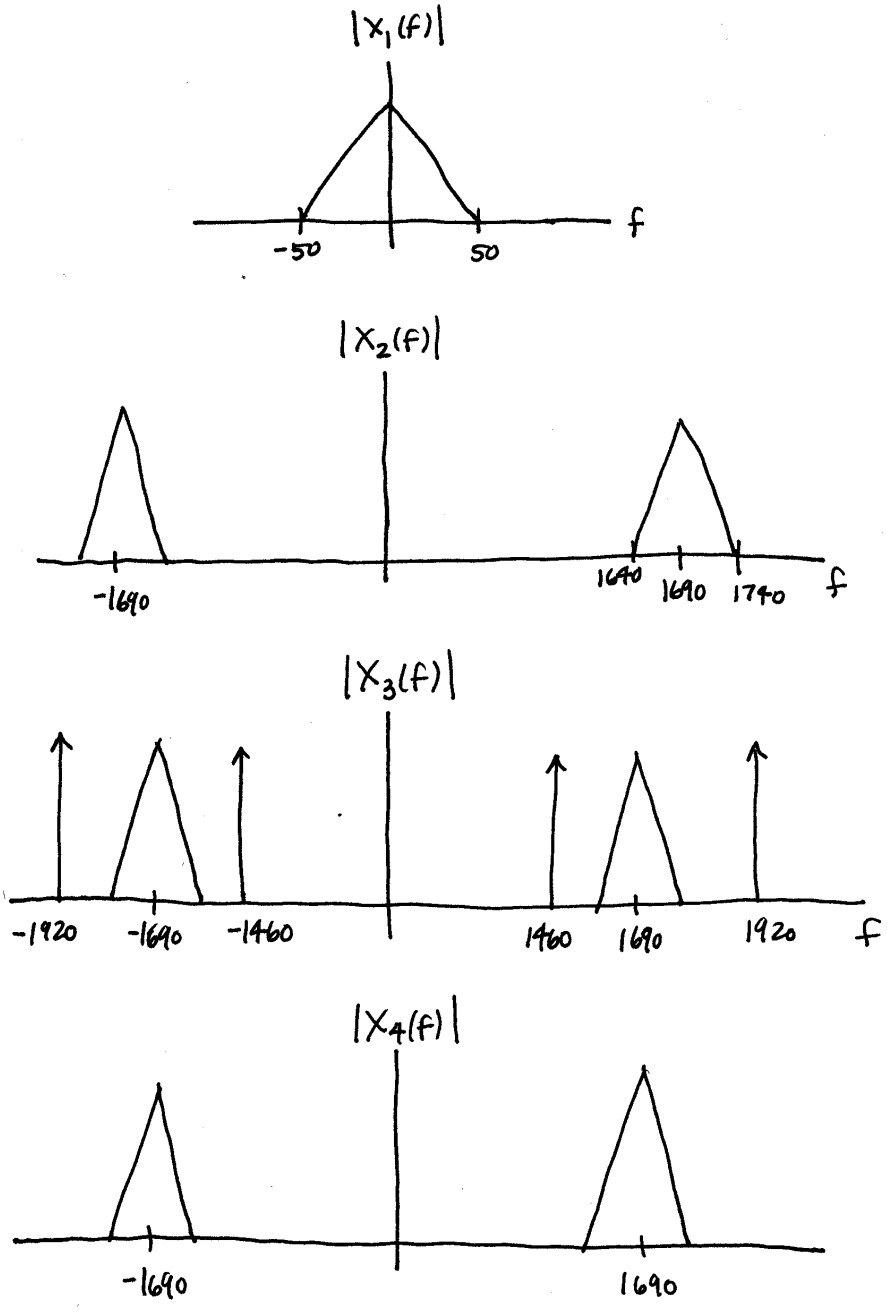
Another Example (cont'd)

- For the following specifications in kHz
  - ⊙ $f_1 = 50$
  - ⊙ $f_2 = 1690$
  - ⊙ $f_3 = 1920$
  - ⊙ $f_4 = 1460$
  - ⊙ $f_5 = 1620$
  - ⊙ $f_6 = 1760$
  - ⊙ $f_7 = 800$
  - ⊙ $f_8 = 90$
  - ⊙ $f_9 = 60$

  given $|X_1(f)|$ as even-symmetric, triangular shaped, and centered at zero frequency and $M = 2$, we can draw $|X_i(f)|$ for $i = 1, 2, ..., 8$ to show that $|X_8(f)|$ matches (up to a scalar gain factor) the magnitude spectrum of $x_1(t)$ sampled at the symbol rate.

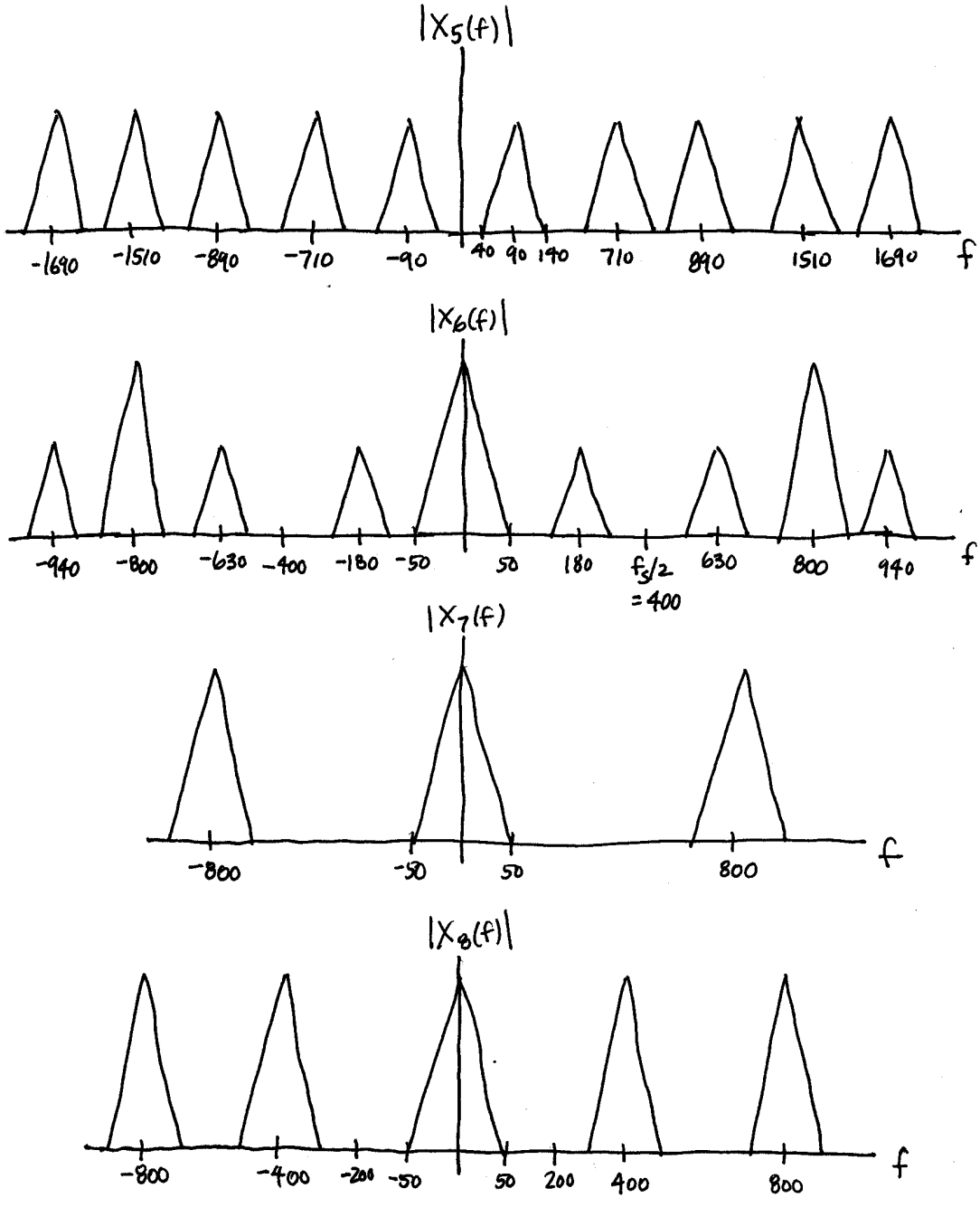
Sub-Nyquist Sampling (cont'd)

Another Example (cont'd)



Sub-Nyquist Sampling (cont'd)

Another Example (cont'd)



Interpolation

- *Objective*: Use signal samples from times kT_s to reconstruct the analog signal value at a time instant not among the set of sample times.
- *Sinc interpolator*:

  $$w(t)|_{t=\tau} = w(\tau) = \int_{\rho=-\infty}^{\infty} w_s(\rho)\text{sinc}(\tau - \rho)d\rho$$

  Because $w_s(\rho)$ is nonzero only when $\rho = kT_s$,

  $$w(\tau) = \sum_{k=-\infty}^{\infty} w_s(kT_s)\text{sinc}(\tau - kT_s)$$

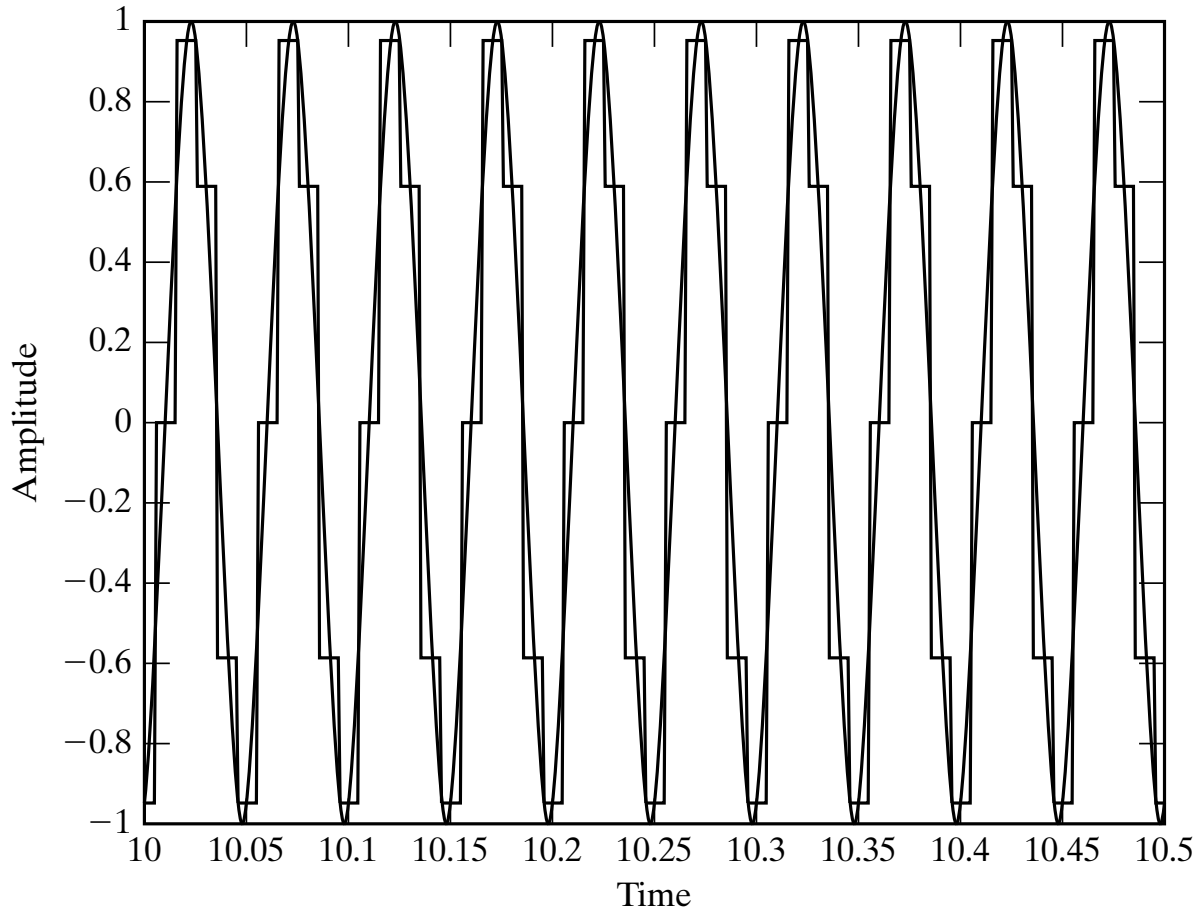
- *Prescription for perfection*: As long as $f_s > 2B$ (where B is the highest frequency present in $w(t)$) this (doubly infinite) sinc interpolator is exact.
- *Filtering interpretation*: Creation of $w(\tau)$ can be interpreted as a convolution of w_s with a sinc-shaped impulse response.

Interpolation (cont'd)

- *Ideal LPF Interpolator*: Convolution in time domain is multiplication in frequency domain. Spectrum of sinc is a rectangle, i.e. an ideal LPF. Thus, an ideal lowpass filter with appropriate cutoff frequency is a perfect interpolator for a Nyquist-sampled signal.
- *Perfection inhibiting practicalities*: In practice, it is necessary to truncate the doubly infinite convolutional sum. Furthermore, $w(t)$ can always be expected to have traces of frequencies above B. Therefore, in practice, we must settle for an approximation.
- *Non-ideal LPF interpolator*: Fortunately, any suitable LPF (with nonzero, flat magnitude and linear phase up to frequency B and fully rejecting before reaching next higher frequency chunk in spectrum of w_s) will provide accurate interpolation.

Interpolation (cont'd)

Example: Using `sininterp` (which uses `interpsinc`) to reconstruct a sinusoid sampled five times per period (as indicated by the choppy staircase zero-order-hold reconstruction of the samples)

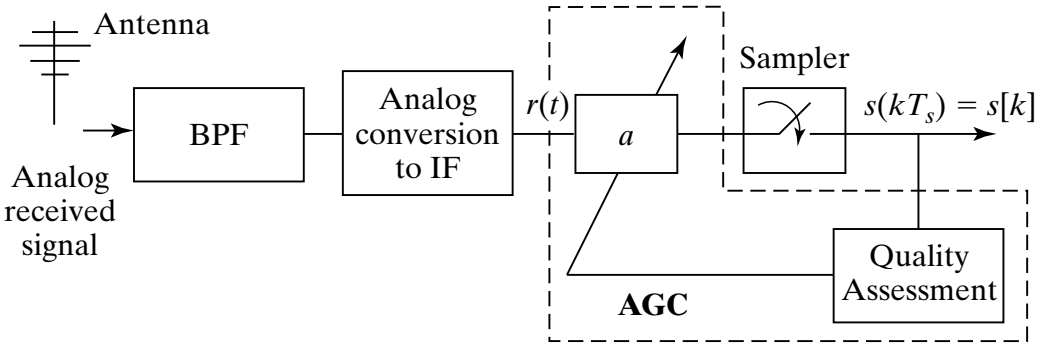


AUTOMATIC GAIN CONTROL

- ⋆ Automatic Gain Control Algorithm Construction
- ⋆ Tracking Example: Time-Varying Fade

Sampling with AGC

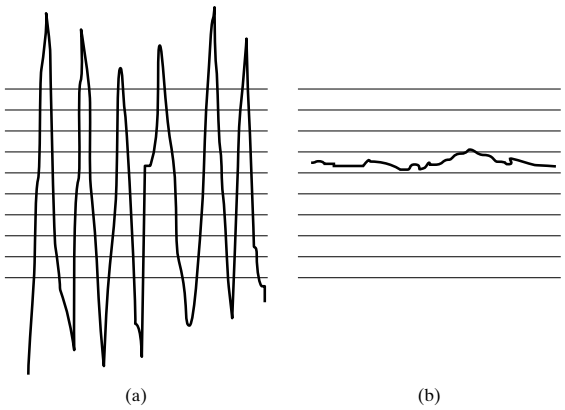
We now focus on the sampler and its surrounding automatic gain control (AGC) in a receiver front end



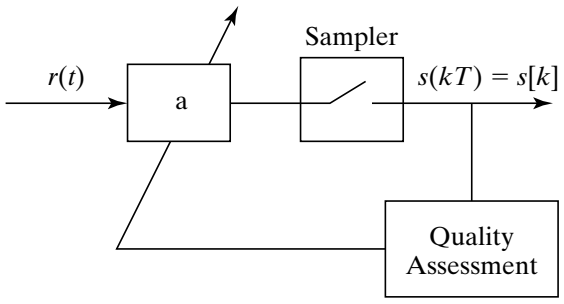
Our purpose here is more to introduce a strategy for parameter adaptation that will be repeated for carrier and clock recovery and equalization, rather than to promote a particular AGC algorithm.

Automatic Gain Control (AGC)

- An AGC maintains the dynamic range of a (zero-average) signal by attenuating when it is too large (as in (a)) and by amplifying when too small (as in (b)).



- AGC adjusts gain parameter a so average energy at output remains (roughly) fixed, despite fluctuations in average received energy.



AGC (cont'd)

Gain Tuning:

- We are to choose a for a received waveform $r(t)$ segment that produces sampler outputs $s[k]$ with the intent of having the average s^2 value over that dataset match a preselected constant d^2.
- Because $s[k] = ar(kT_s)$, we can choose

$$a^2 = \frac{d^2}{\left(\frac{1}{N}\sum_{i=1}^{N} r^2[k+i]\right)} = \frac{d^2}{\text{avg}\{r^2[k]\}}$$

  (preferring $a > 0$) to make (as desired)

$$\{\frac{1}{N}\sum_{i=1}^{N} s^2[k+i]\} = d^2$$

- Unfortunately, we need the samples of r, which are not available on the DSP side of the receiver, to solve this formula for a.
- Our search for a gain tuner continues.

AGC (cont'd)

Heuristic Algorithm Development:

As an alternative, consider the following strategy:

- select an initial positive a.
- As a sample s arrives, compare its square to d^2.
- If s^2 at that particular sample instant is greater than d^2, we will reduce a positive a to a smaller positive value. If a is negative, we would decrease its magnitude, i.e. increase it toward zero.
- Plus, the correction term should be larger the further d^2 is from s^2.
- Similarly, if $s^2 < d^2$, we will increase a positive a by an amount proportional to $d^2 - s^2$. If a is negative, a should be decreased (i.e. made more negative), so its magnitude increases.

AGC (cont'd)

An algorithm that performs this strategy is

$$a[i+1] = a[i] + \mu\{\text{sign}(a[i])\}(d^2 - s^2[i])$$

where μ is a suitably small positive stepsize. (The sign($a[i]$) term can be removed if $a[i]$ starts and stays positive.)

- Can this algorithm be implemented from data available on the DSP side of the sampler?

  Ans: Yes, s (and not r) is needed

- Will this algorithm converge to the desired a of $\pm d/\sqrt{\frac{1}{N}\sum_{i=1}^{N} r^2[i]}$?

  Ans: It depends what you mean by "converge".

AGC (cont'd)

- The candidate algorithm

$$a[i+1] = a[i] + \mu\{\text{sign}(a[i])\}(d^2 - s^2[i])$$

  cannot be expected to converge to a fixed value.

- Because r ranges widely, only on average does a^2r^2 (or s^2) actually equal d^2.

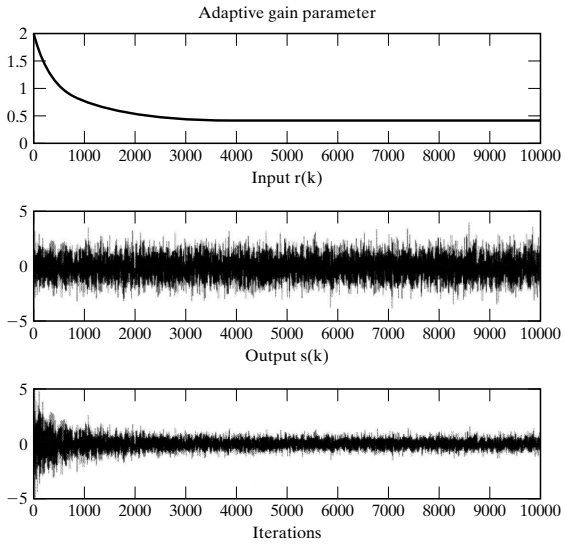
- The resulting (typically) nonzero instantaneous error in $d^2 - s^2$ and a nonvanishing stepsize μ will result in a change in a even if it is already at the right value for the average behavior of s^2.

- A sufficiently small μ should keep this asymptotic rattling within a tolerable level.

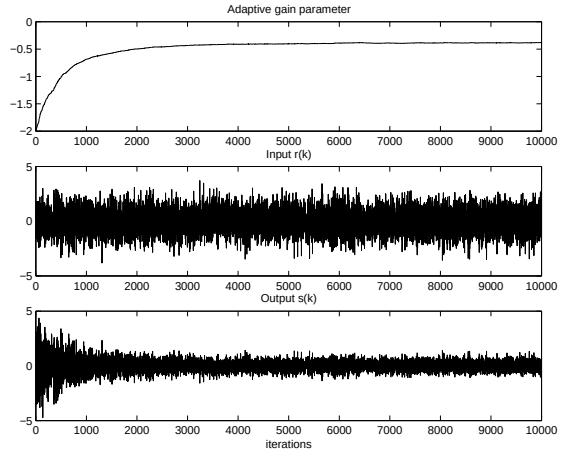
AGC (cont'd)

Testing:

- Using `agcgrad` with $\text{avg}\{r^2\} \approx 1$ and $d^2 = 0.15$, the desired $a \approx \sqrt{0.15} \approx 0.38$.
- Start at $x = 2$ with $\mu = 0.001$

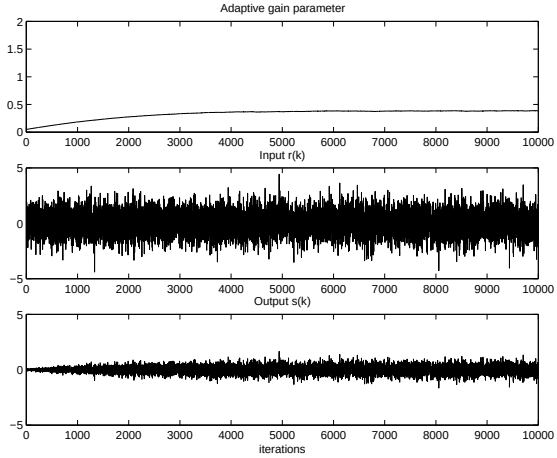


- Start of $x = -2$ with $\mu = 0.001$

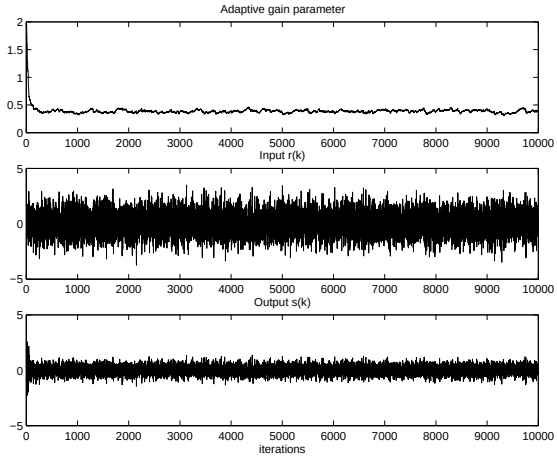


AGC (cont'd)

- Start at $x = 0.05$ with $\mu = 0.001$



- Start at $x = 2$ with $\mu = 0.02$



AGC (cont'd)

Observations:

- Asymptotically, this algorithm hovers in a small region about the desired answer.
- The asymptotic hovering region's size can be decreased by reducing the stepsize μ, which also reduces the algorithm convergence rate.
- When the average value of the hovering parameter has effectively reached a fixed value, the average of $a[i+1]$ will equal the average of $a[i]$ such that from our algorithm

$$a[i+1] = a[i] + \mu \text{sign}(a[i])(d^2 - s^2[i])$$

  the average of the correction term $\mu \text{sign}(a[i])(d^2 - s^2[i])$ must be zero.
- With $\mu > 0$ and the asymptotic hovering $a[i]$ not changing sign, zeroing the average correction term zeros the average of $d^2 - s^2$. But, indeed that is what we seek.

AGC (cont'd)

Gradient Descent Algorithm Development:

- As a more generalizable approach to adaptor algorithm development consider specifying a cost function and using an iterative optimizer based on gradient descent

$$a[i+1] = a[i] - \mu \frac{\partial J_N(a)}{\partial a}|_{a=a[i]}$$

- Try $J_N(a) = \text{avg}\{|a|((s^2[k]/3) - d^2)\}$ with the definition of "avg" as

$$\text{avg}\{x[k]\} = (1/N) \sum_{i=k}^{k-N+1} x[i]$$

- For small stepsize μ, differentiation and averaging are approximately interchangeable

$$\frac{\partial J_N(a)}{\partial a} = \frac{\partial}{\partial a}[\text{avg}\{|a| \left(\frac{a^2 r^2 (kT)}{3} - d^2 \right)\}]$$

$$\approx \text{avg}\{\frac{\partial}{\partial a}[|a| \left(\frac{a^2 r^2 (kT)}{3} - d^2 \right)]\}$$

AGC (cont'd)

- With $\frac{\partial|a|}{\partial a} = \text{sign}(a)$ and $\frac{dw}{dx} = \frac{dw}{dy} \cdot \frac{dy}{dx}$

$$\frac{\partial J_N(a)}{\partial a} \approx \text{avg}\{|a|(1/3)2ar^2(kT)$$

$$+\text{sign}(a)(1/3)a^2r^2(kT)\} - \text{sign}(a)d^2$$

- With $\text{sign}(a)|a| = a$

$$\frac{\partial J_N(a)}{\partial a} \approx \text{avg}\{\text{sign}(a)\left(a^2r^2(kT) - d^2\right)\}$$

- With $a^2r^2 = s^2$

$$\frac{\partial J_N(a)}{\partial a} \approx \text{avg}\{\text{sign}(a)\left(s^2[k] - d^2\right)\}$$

  So, the stationary points of zero gradient are in the right places with $\text{avg}\{s^2\} = d^2$.

- With $\partial(\text{sign}(a))/\partial a = 0$ everywhere but $a = 0$, the second derivative is approximately

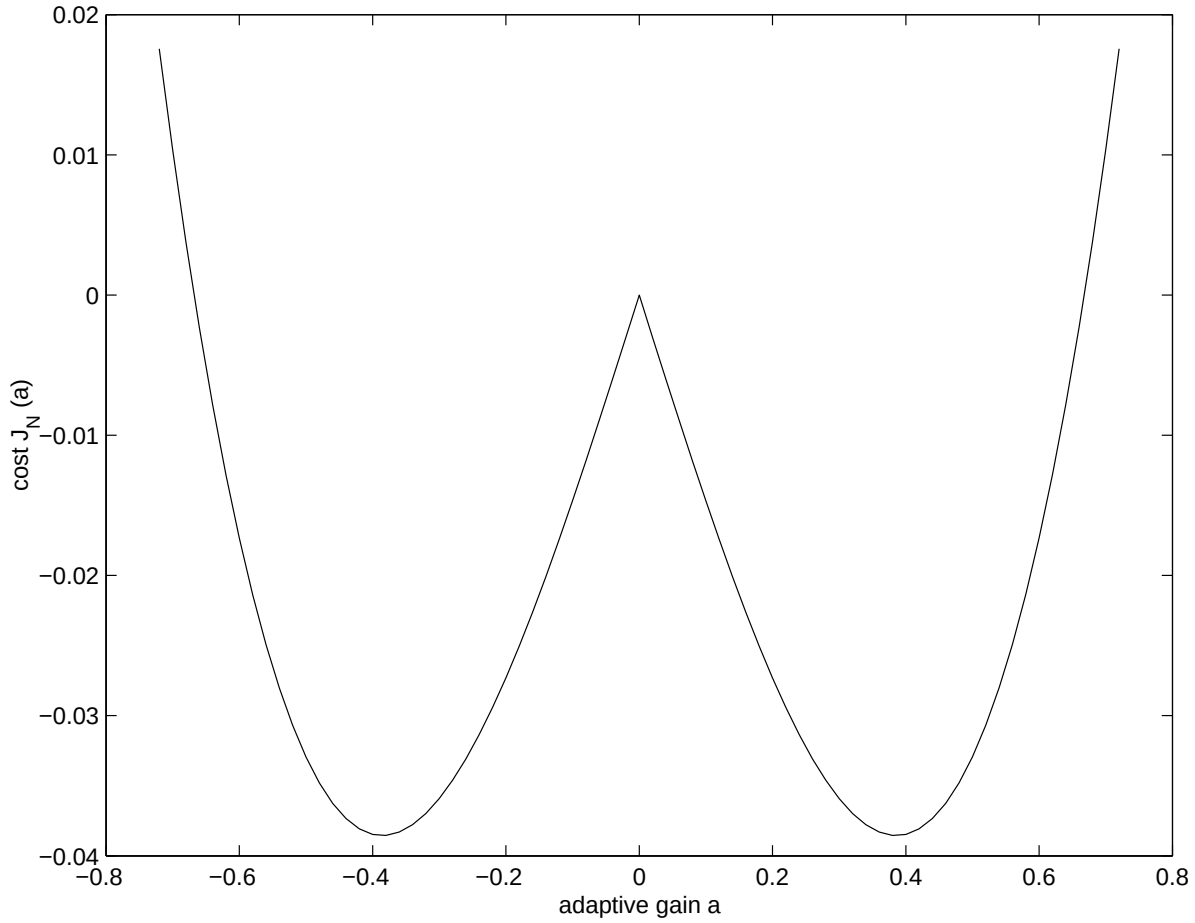
$$\text{avg}\{\frac{\partial}{\partial a}\left[\text{sign}(a)\left(a^2r^2(kT) - d^2\right)\right]\}$$

  $= \text{avg}\{2a\,\text{sign}(a)r^2(kT)\} = \text{avg}\{2|a|r^2(kT)\} > 0$

  So, stationary points at $a \neq 0$ are minima.

AGC (cont'd)

- With constant avg$\{r^2\}$ and d, J_N has double dip "egg carton" style cross section
- For specific data set (with $N = 1000$) from `aes`



AGC (cont'd)

- Computation of the gradient requires that a remain constant over the N samples over which avg$\{s^2\}$ is composed.
- Consider squeezing the averaging window to a single sample so $N = 1$ and

$$a[i+1] = a[i] - \mu \text{sign}(a[i]) \left(s[i]^2 - d^2 \right)$$

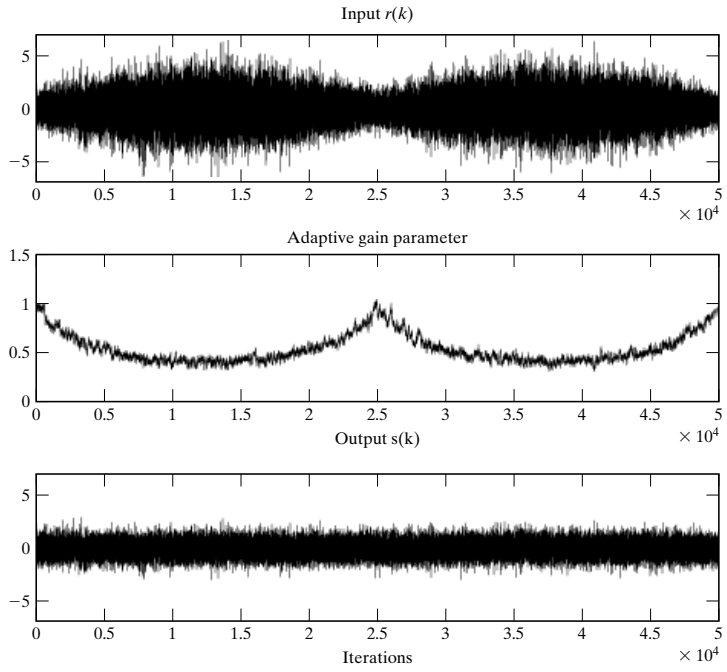
- This is the algorithm developed heuristically and tested previously.
- This algorithm also emerges from first reducing the averaging window to $N = 1$ in the cost function and then taking the gradient and forming a gradient descent iteration.
- This technique of shrinking the averaging window so averaging is explicitly removed works because LPF action of adaptation acts similarly to averaging before updating.

Tracking Example: Time-Varying Fade

- To demonstrate desired tracking capability, use `agcvsfading` to test

$$a[i+1] = a[i] - \mu \text{sign}(a[i]) \left(s[i]^2 - d^2 \right)$$

with $\mu = 0.01$, $d^2 = 0.5$, $a[1] = 1$, and a large, slow, oscillating channel gain



- Fade must be changing sufficiently slowly and the input must never die for the AGC with small stepsize to track adequately.

Laboratory Exercises – Day 1

Introduction to Digital Communication Receiver Design

Task 1: Filter Design With `remez`

The MATLAB command `remez` is useful for generating so-called "equiripple FIR filters". We will rely on it frequently for designing lowpass and bandpass filters. The `remez` command takes three parameters. Type `help remez` to familiarize yourself with the parameters – you only need to pay attention to the first paragraph in the help, called with 3 parameters N,F, and A.

The following code generates 3 seconds worth of a random (white) signal sampled at 10 kHz, and plots the magnitude spectrum:

```
time=3;
Ts=1/10000;
x=randn(time/Ts,1);
plotspec(x,Ts);
```

The following lines design a 100-th order low-pass filter with a cutoff at 1 kHz, and plots the filtered signal:

```
h=remez(100,[0 0.2 0.21 1],[1 1 0 0])';
y=filter(h,1,x);
plotspec(y,Ts);
```

Your task: Provide the corresponding lines of code to design a bandpass filter (BPF) which passes frequencies between 1.5 kHz and 2.5 kHz. Plot the result of filtering `x` with the BPF. Plot the result of filtering `y` with the BPF.

Task 2: Filtering with Tapped Delay Lines

The `filter` and `conv` commands are quite useful for filtering signals, but they assume you have all of the data available. In a real-time communication system, we may want to put each sample into a filter as we receive it. In this case, the `filter` and `conv` commands are not so useful. For a signal $x[n]$ passing through a filter $h[n]$ of length N, the output at time n is given by the convolution sum:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

We can implement the convolution sum very efficiently in MATLAB using vector inner products. For example, the filter output at time n is given by

```
y(n)=h'*x(n:-1:n-N);
```

Your task: Using `for` loops and vector inner products, write a few lines of code that are equivalent to the command `y=filter(b,1,x)`. Compare your result with the the previous problem where you used the `filter` command, and calculate the mean squared error between the two (Note: you may ignore the first N samples in the error calculation, where N is the number of taps in the filter).

Task 3: Detection via Correlation

In packet-based wireless communication systems, the beginning of the transmission usually contains a marker sequence. The receiver is constantly looking for such a marker sequence; when it detects that a marker sequence has been sent, it knows that data is about to be transmitted, and it knows the location of the "start" of the packet.

The standard technique for identifying a marker sequence is called correlation. Correlation is much like convolution, but with a sign change in the indexing. If $y[n]$ is the received signal, $marker[n]$ is the (known) marker sequence of length N, the correlator output z at time n is given by

$$z[n] = \sum_{k=0}^{N-1} marker[k]y[n+k].$$

When the correlator output $z[n]$ exceeds some pre-determined threshold, the receiver decides that the marker was identified at that value of n.

Your task: Load the file `/day1/correl_ex.mat` by typing `load correl_ex`. This file contains two variables: a length 100 marker sequence called `marker`, and a length 2000 received sequence called `y`. Write a few lines of code to perform the correlation and determine the starting location of the marker sequence. Also, show a plot of $z[n]$.

Task 4: Amplitude Modulation

Consult the file `/day1/AM.m`. This code generates a message $w(t)$ and modulates it with a carrier at frequency f_c. The demodulation is done with a cosine of frequency $f_c + \gamma$ and a phase offset of ϕ. When $\gamma = 0$ and $\phi = 0$ (i.e. in the ideal conditions), the output is identical to the original message, except for the inevitable delay caused by the linear filter.

Your tasks:

1. Plot the signals $w(t)$, $v(t)$, $x(t)$, and $m(t)$, and describe what you see.
2. Using the `plotspec` command, plot the spectra of these same signals. Describe what you see.
3. Change the phase offset, ϕ. Describe the effect for different values.
4. Change the frequency offset, γ. Describe the effect for different values.

Task 5: Sinc Interpolation

As you should be aware, sampling a signal faster than the *Nyquist rate* allows for perfect reconstruction since no information is lost. However, once we have a sampled digital signal, how do we reconstruct the data between samples? The answer is sinc interpolation.

We will use sinc interpolation quite often in our digital receiver, particular during baud-timing. The function `/day1/interpsinc.m` performs sinc interpolation, and we will use this frequently. Open this file, and familiarize yourself with its operation.

To see an example of using sinc interpolation, consider interpolating the points of a sampled sinusoid. The file `/day1/interp_example.m` generates a sine wave $w(t)$ of frequency 20 Hz with a sampling rate of 100 Hz. The code then shows how to use `interpsinc.m` to interpolate between the samples.

Your task: Generate a new wave $w(t)$ which is the sum of 2 sinusoids – one with frequency 17 Hz, and one with frequency 20 Hz. Consider t between -10 and 10. Let $w(kT_s)$ represent samples of $w(t)$ with $T_s = 0.01$. Use `interpsinc.m` to interpolate the values $w(0.011)$, $w(0.013)$, and $w(0.015)$, using $10\times$ oversampling. Compare the interpolated values to the actual values.

Task 6: Automatic Gain Control via Gradient Descent

The function `/day1/agcgrad.m` implements the AGC gradient descent algorithm which minimizes the cost

$$J_N(a) \quad = \quad \text{avg}\left\{|a|\left(\frac{a^2r^2}{3} - ds\right)\right\}$$

by choice of a. The gain parameter a adjusts automatically to make the overall power of the output s roughly equal to the specified parameter ds. Run `agcgrad.m` and you will see that a converges to about 0.38 since $0.38^2 \approx 0.15 = ds^2$.

Your task: Using `agcgrad.m`, answer the following questions

1. What range of stepsize `mu` works? What happens if it is too small? too large?
2. How does choice of `mu` effect convergence rate?
3. How does the variance of the input effect the convergent value of a?
4. Try initializing the estimate `a(1)=-2`. Which minimum does the algorithm find? What happens to the data record?

DAY 2

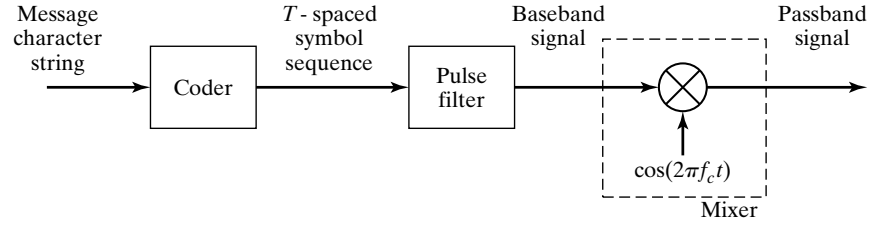
- RF System Simulation with Impairments
- Carrier Recovery

AN IDEALIZED RF SYSTEM SIMULATION

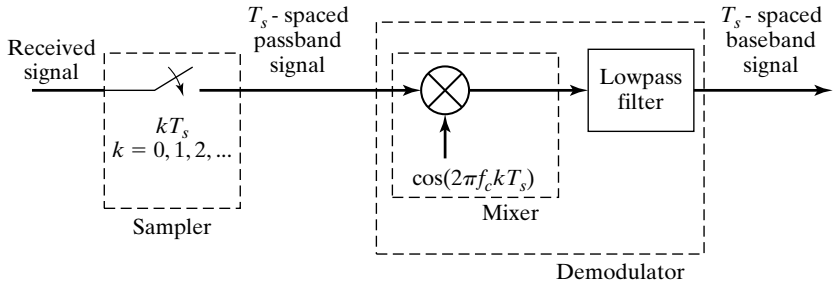
- ⋆ A Naive/Ideal Communication System
- ⋆ Flat Fading
- ⋆ What if ...

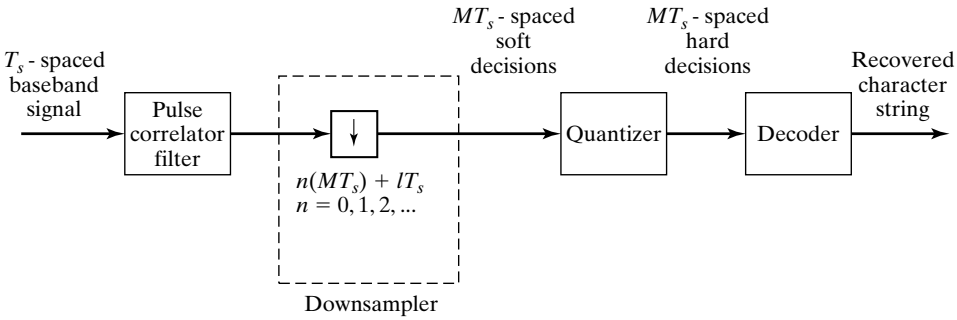
A Naive/Ideal Communication System

With a perfect (i.e. gain with delay) channel and satisfactory carrier, baud timing, and frame synchronization, we simulate this PAM system (using `idsys`).



(a) Transmitter





(b) Receiver

A ... System (cont'd)

TRANSMITTER

- *text message*: 01234 I wish I were an Oscar Meyer wiener 56789
- *coding*: text characters via 8-bit ASCII to 4-PAM $m[i]$
- *baud interval*: $T = 1$ time unit
- *pulse shape*: T-wide Hamming blip $p(\cdot)$
- *carrier frequency*: $f_c = 20$
- *carrier phase*: 0

RECEIVER

- *sampler period*: T_s $(= T/M)$
- *oversample rate*: $M = 100$

A ... System (cont'd)

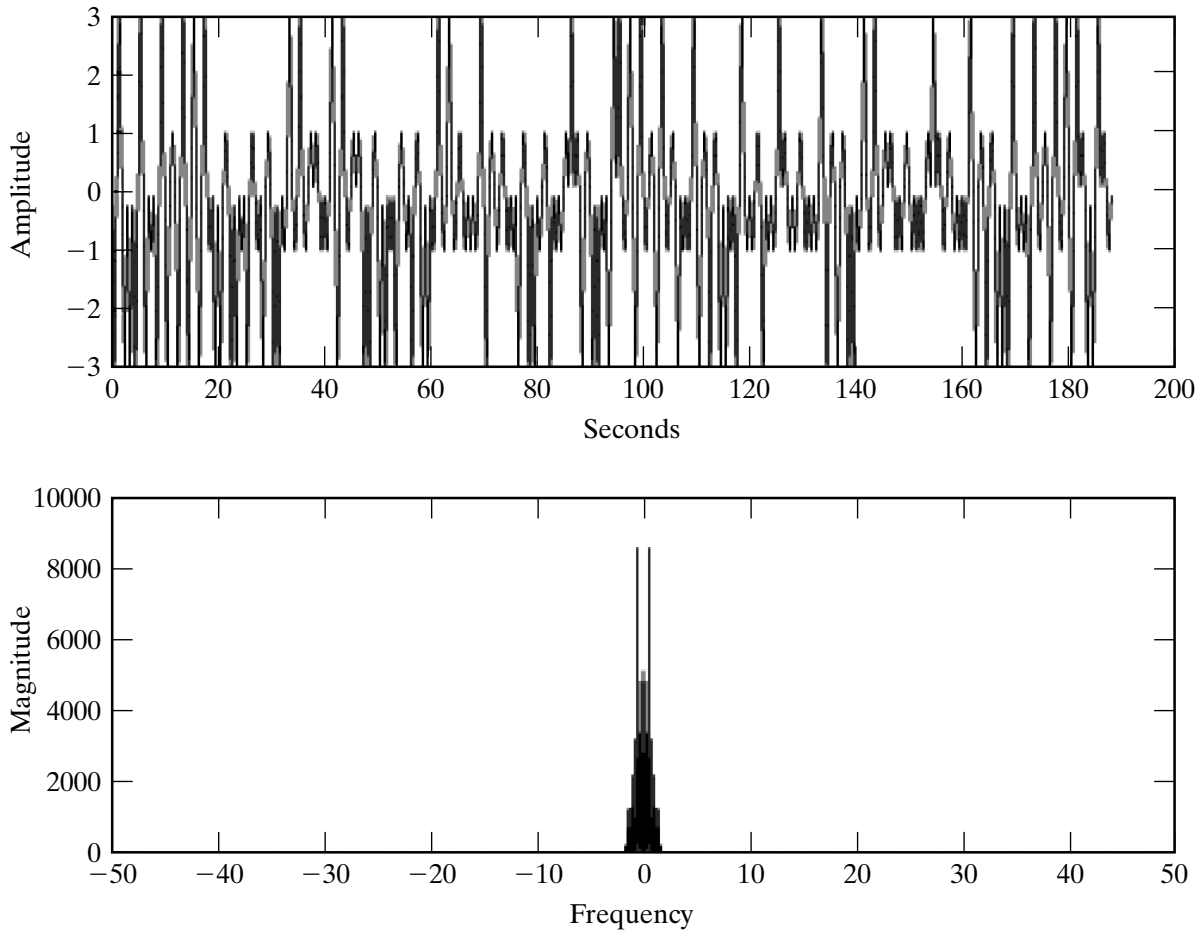
- *free running sampler output*:

$$r(t)|_{t=kT_s} = \sum_{i=0}^{N-1} m[i]p(kT_s - iT)\cos(2\pi f_c kT_s)$$

- *mixer frequency*: $f_c = 20$
- *mixer phase*: 0
- *demodulator LPF*: `remez(fl,fbe,damps)` with `fl` = 50, `fbe` = [0 0.5 0.6 1], and `damps` = [1 1 0 0]
- *pulse correlator filter*: T-wide Hamming blip
- *downsampler baud timing*: $\ell = 125$ (determined experimentally)
- *quantizer*: to nearest element in $\{\pm 1, \pm 3\}$
- *decoder*: 4-PAM to 8 bits via reverse ASCII to text (with frame synchronization assured by indexing from first symbol set by baud timing)

A ... System (cont'd)

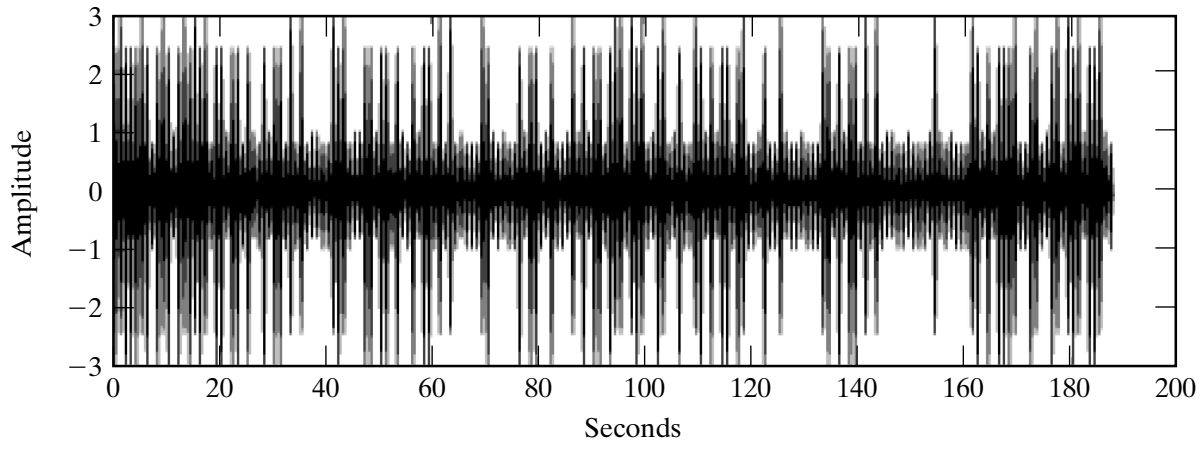
Transmitter baseband signal and magnitude spectrum

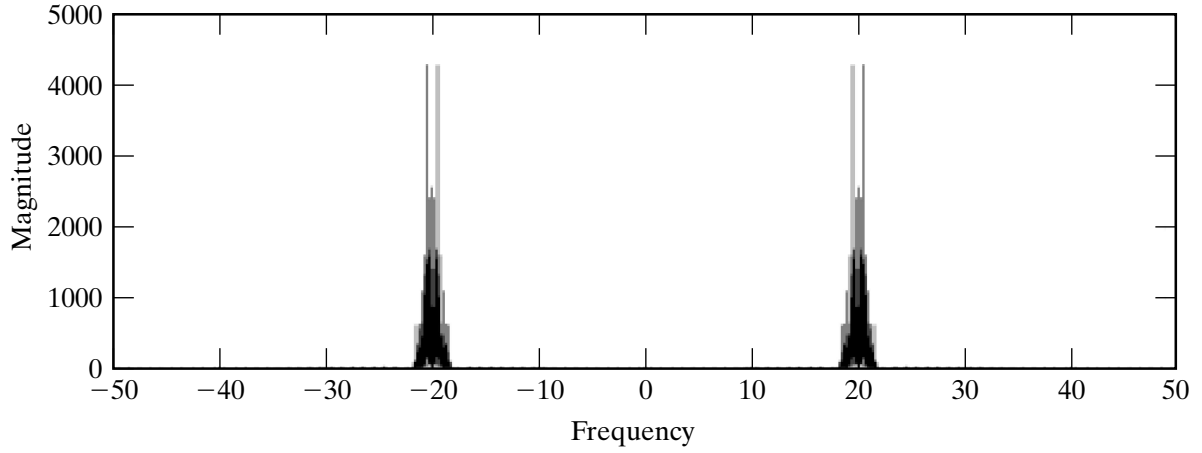


Note that spectrum is limited to minus to plus Nyquist frequency, i.e. half of oversample frequency.

A ... System (cont'd)

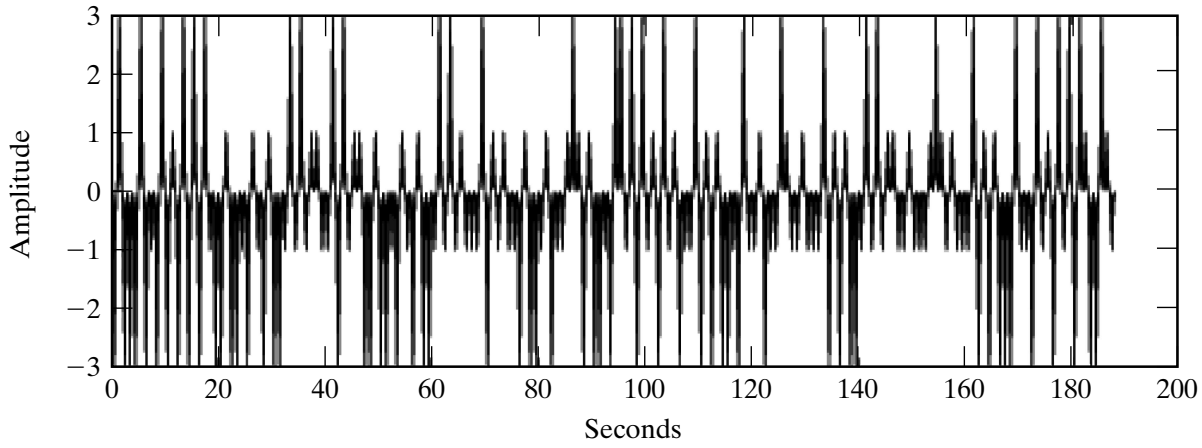
Transmitter passband signal and magnitude spectrum

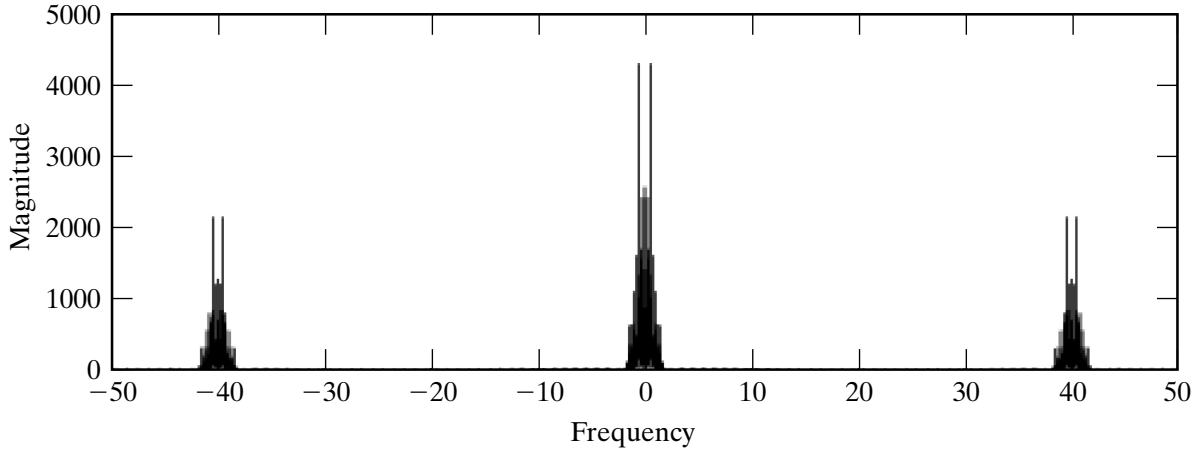




A ... System (cont'd)

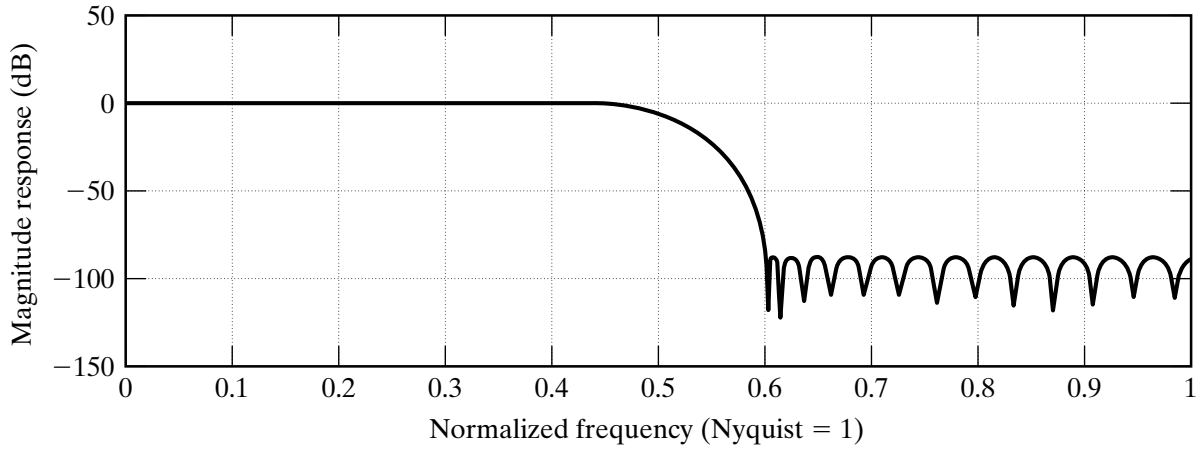
Receiver mixer output and magnitude spectrum

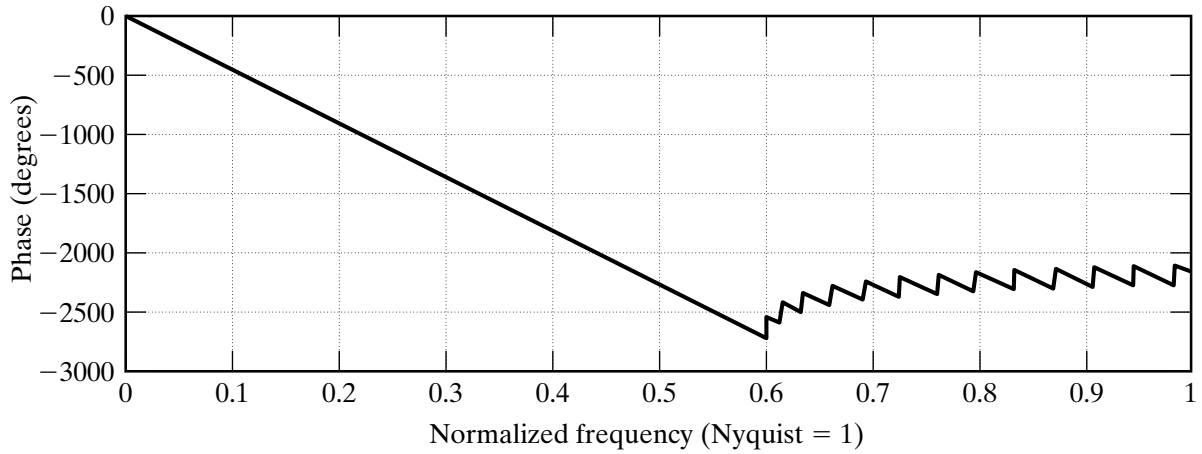




A ... System (cont'd)

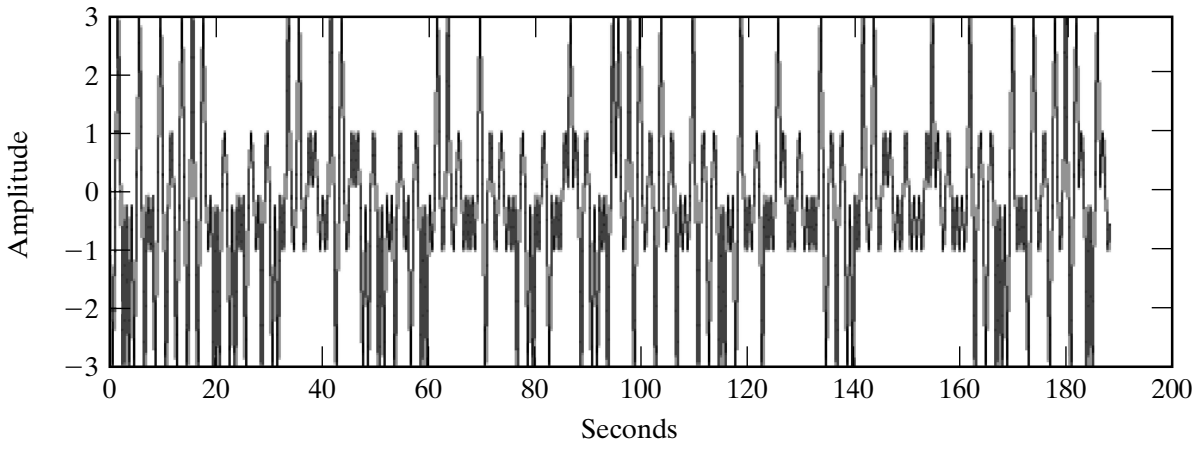
Receiver post-mixer LPF frequency response

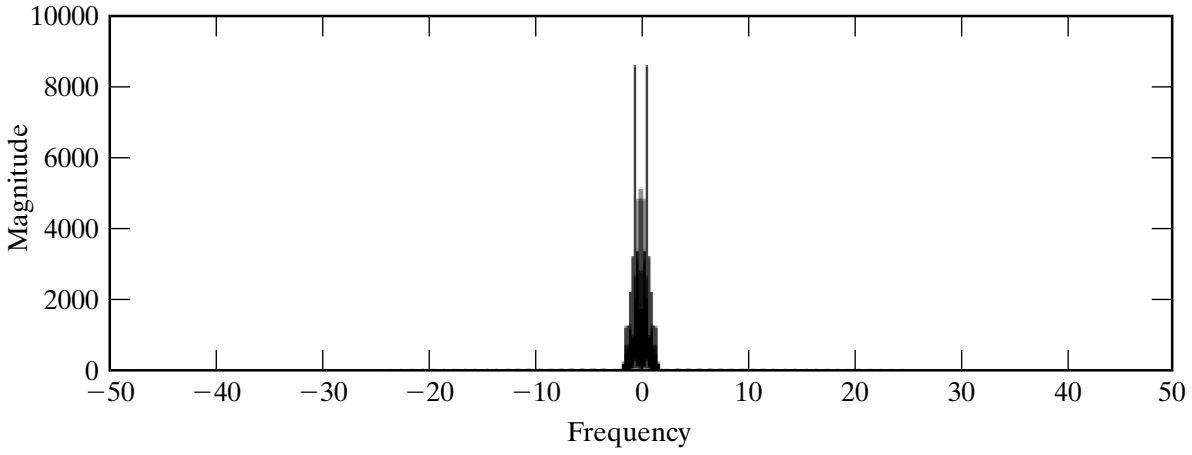




A ... System (cont'd)

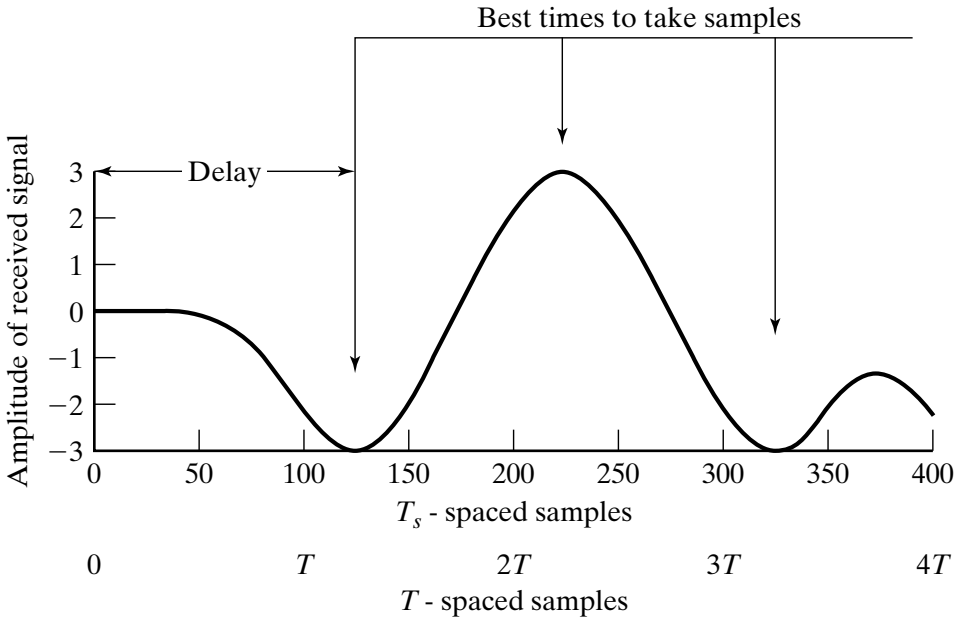
Receiver downconverter-LPF output and magnitude spectrum





A ... System (cont'd)

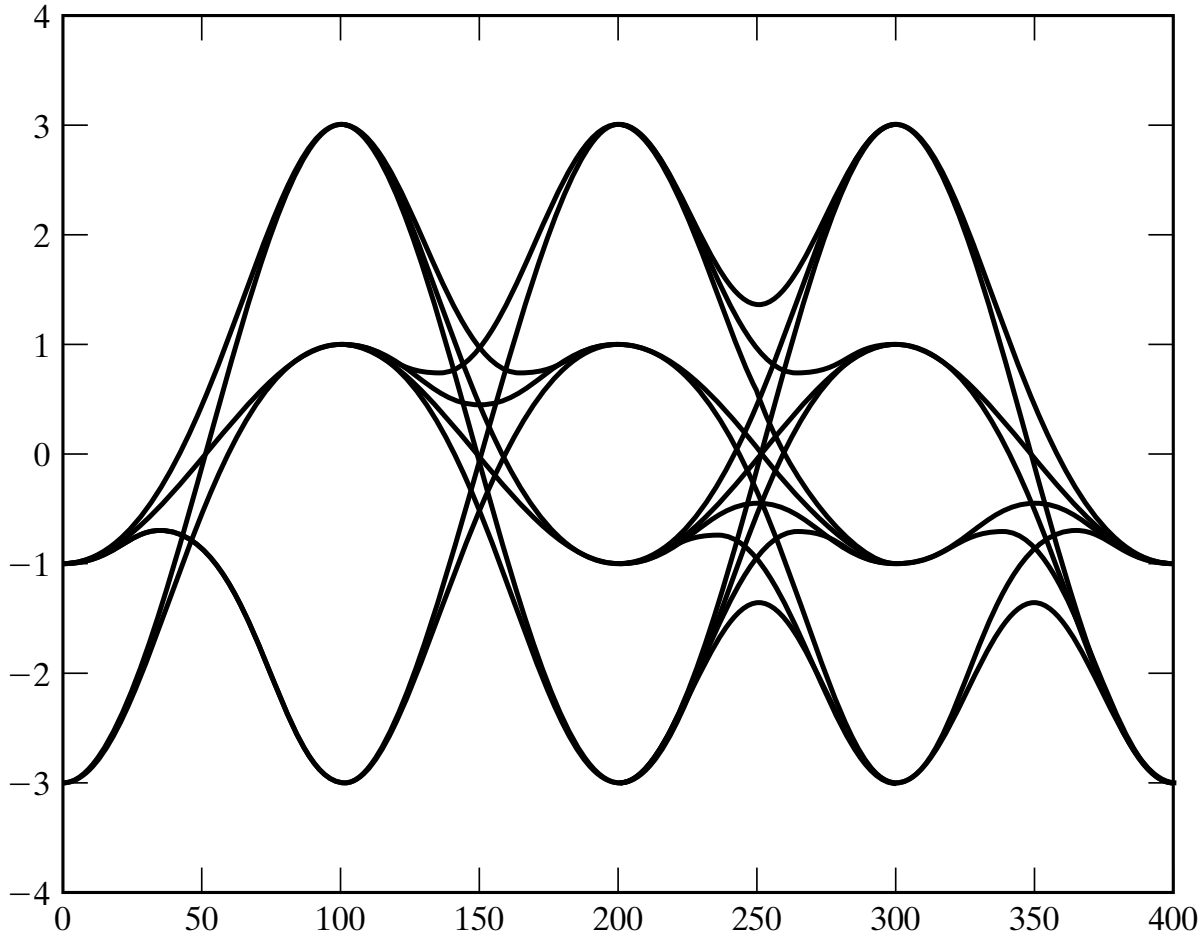
First 400 samples of pulse correlator filter output



This reveals $\ell = 125$ for first symbol sample (or baud) time. (125 = half length of lowpass filter in downconverter and half length of correlator filter and half a symbol period)

A ... System (cont'd)

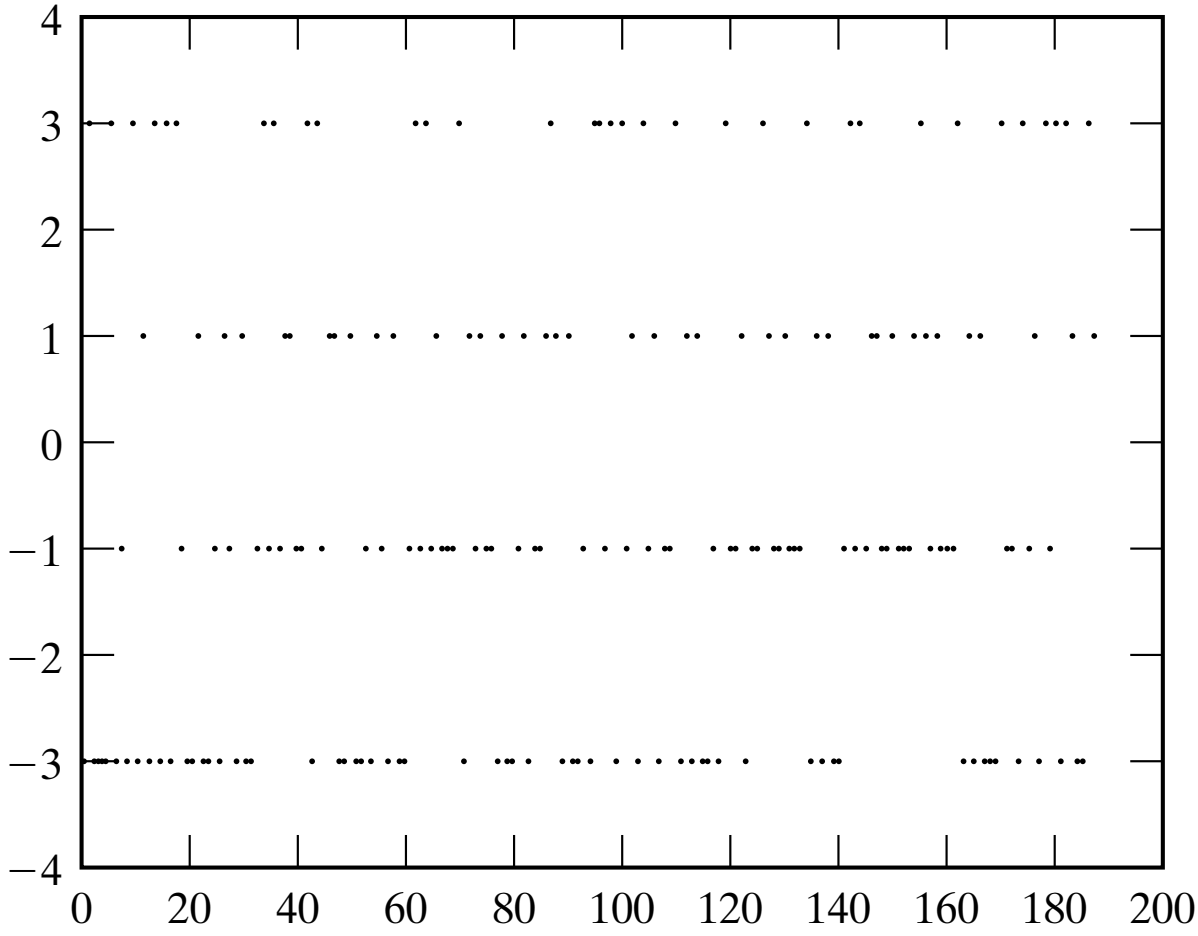
Overlay of successive $4T$-wide correlator output segments starting on first baud time



Note recurrence of pulse peaks at successive T-wide intervals.

A ... System (cont'd)

Soft Decisions Constellation Diagram History

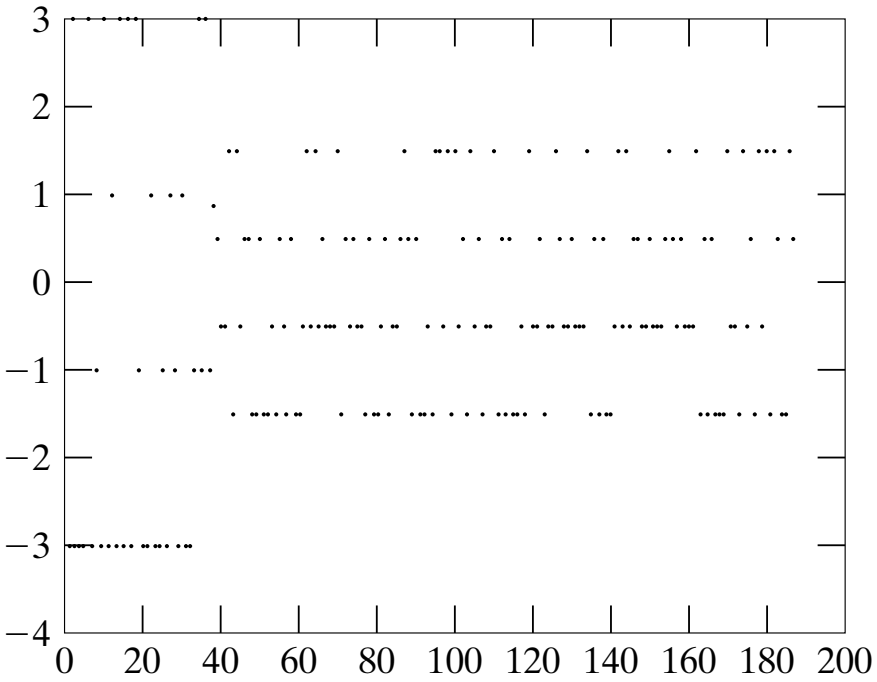


Because the soft decisions are so close to the alphabet levels, there are no decision errors and no symbol errors.

Flat Fading

Impairment: At time representing 20% of duration of simulation window, the channel gain changes abruptly from 1 to 0.5. (as in `idsys+agc`)

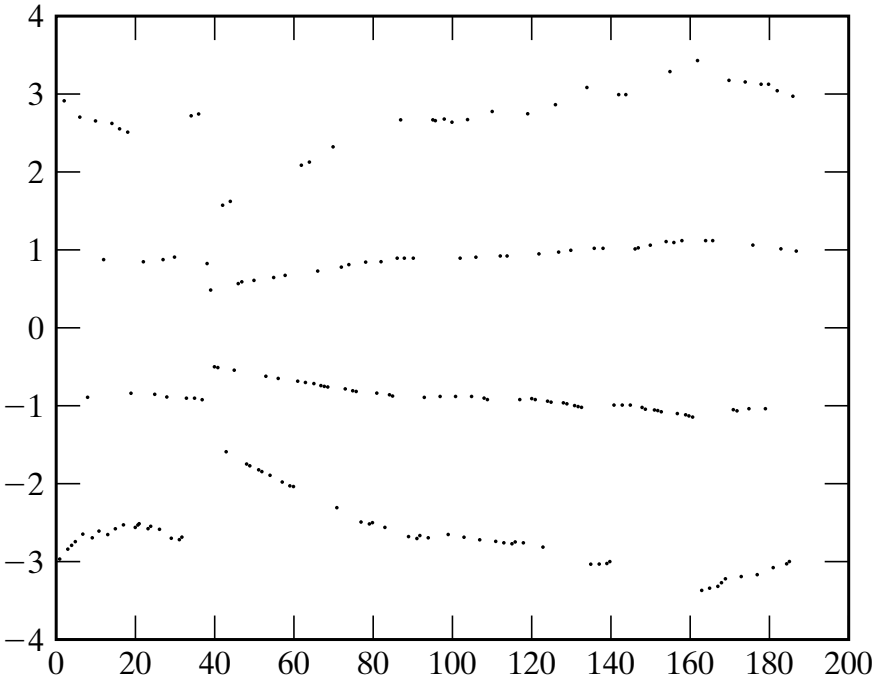
Effect: Soft decisions in "ideal" system receiver



The soft decisions have all moved inside 2 in magnitude, meaning that decision device will never produce $\pm 3 \Rightarrow$ lots of errors.

Flat Fading (cont'd)

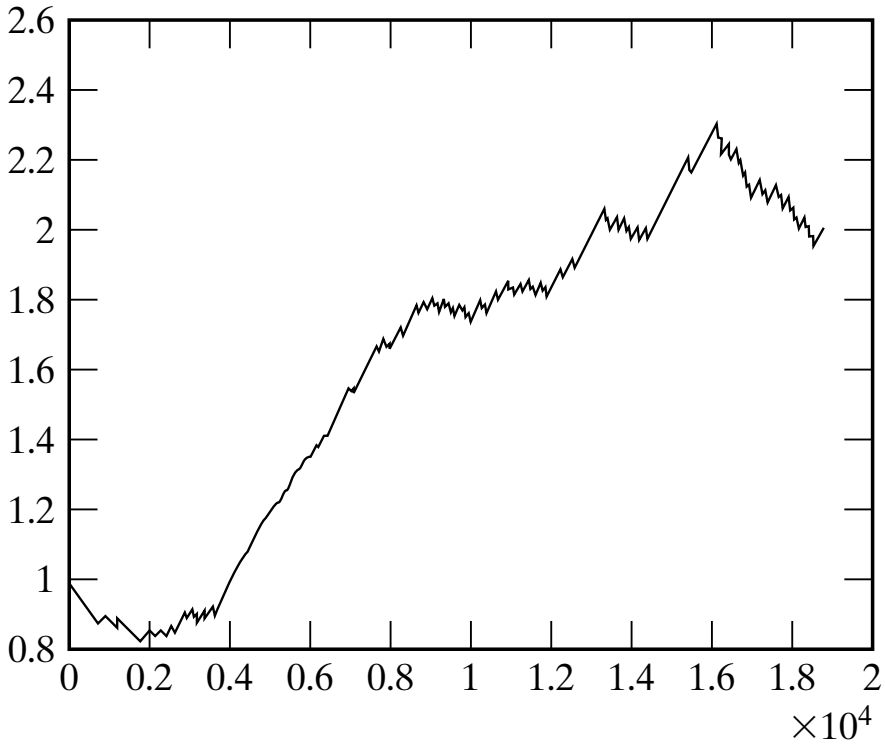
Fixed: Soft decisions with inclusion of AGC



Decisions correct once top and bottom stripes in constellation diagram history have magnitude > 2.

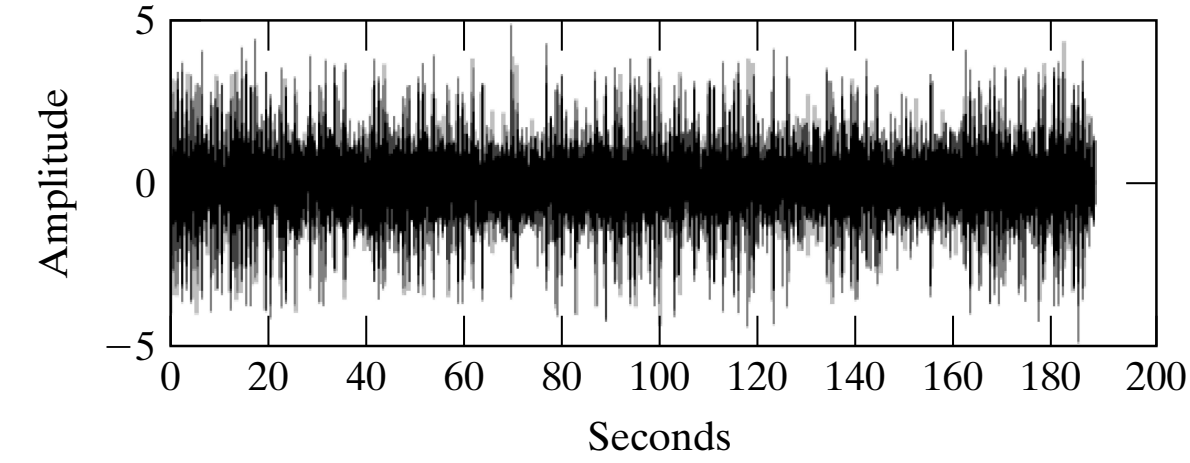
Flat Fading (cont'd)

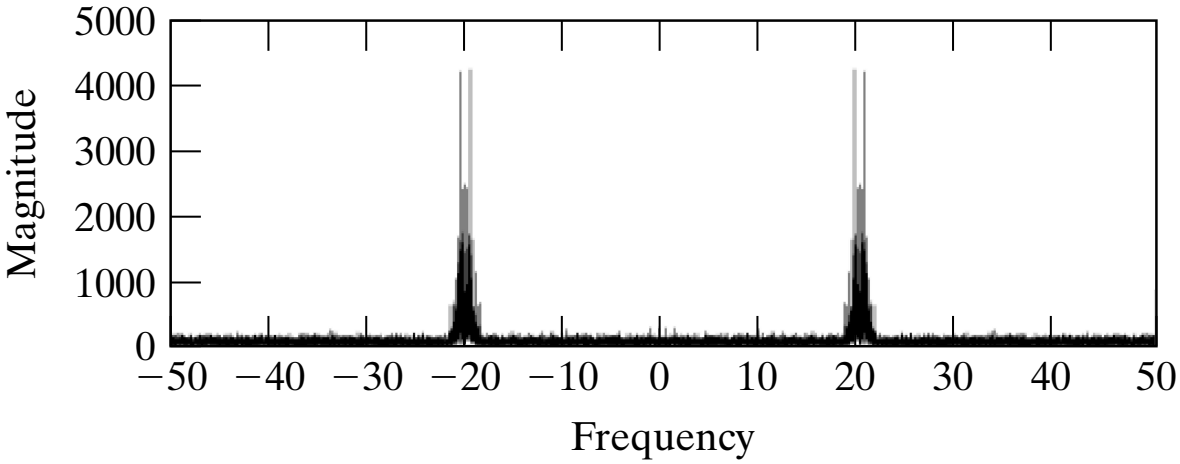
Adapted gain time history: Starts at 1; ends near 2.



What if ...

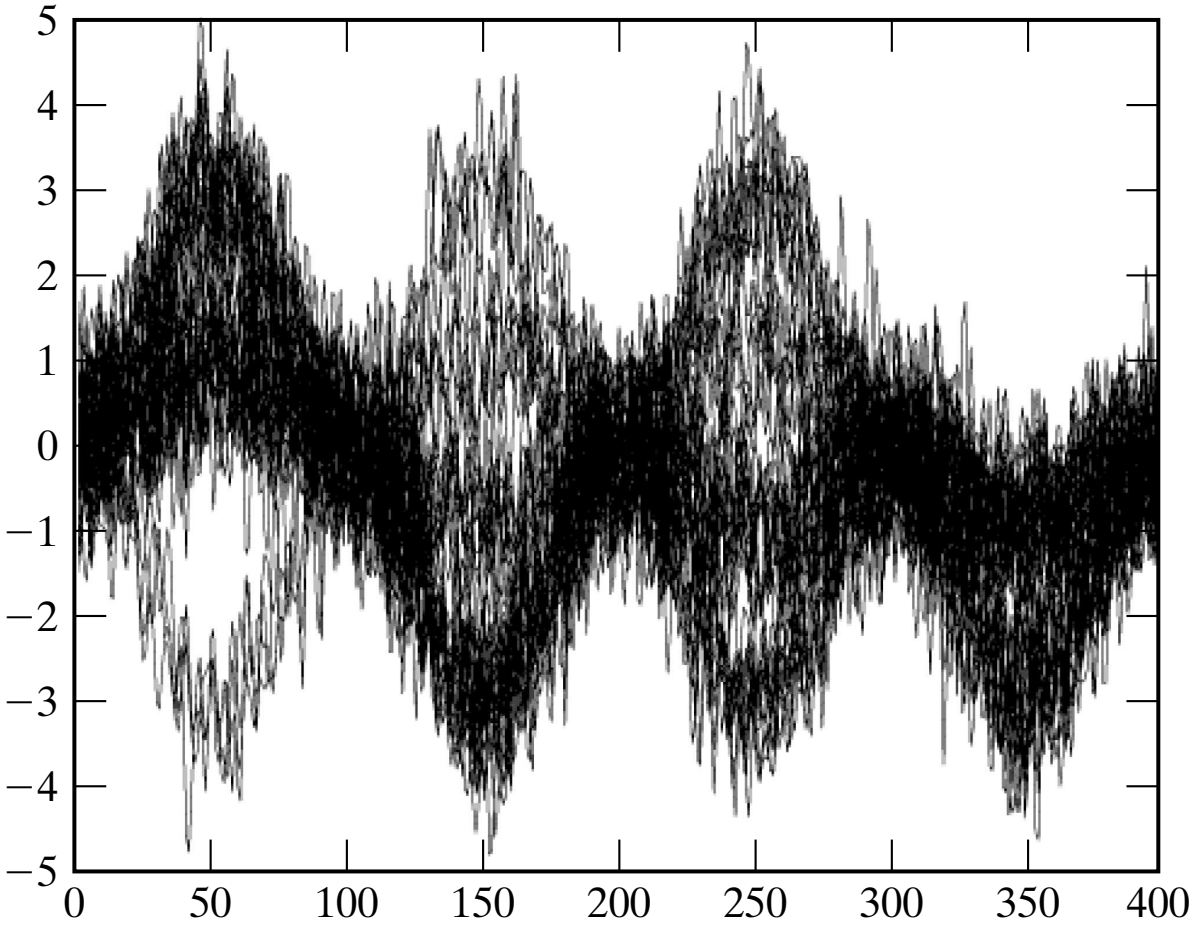
Channel noise: Noisy received signal and spectrum (from `impsys`)





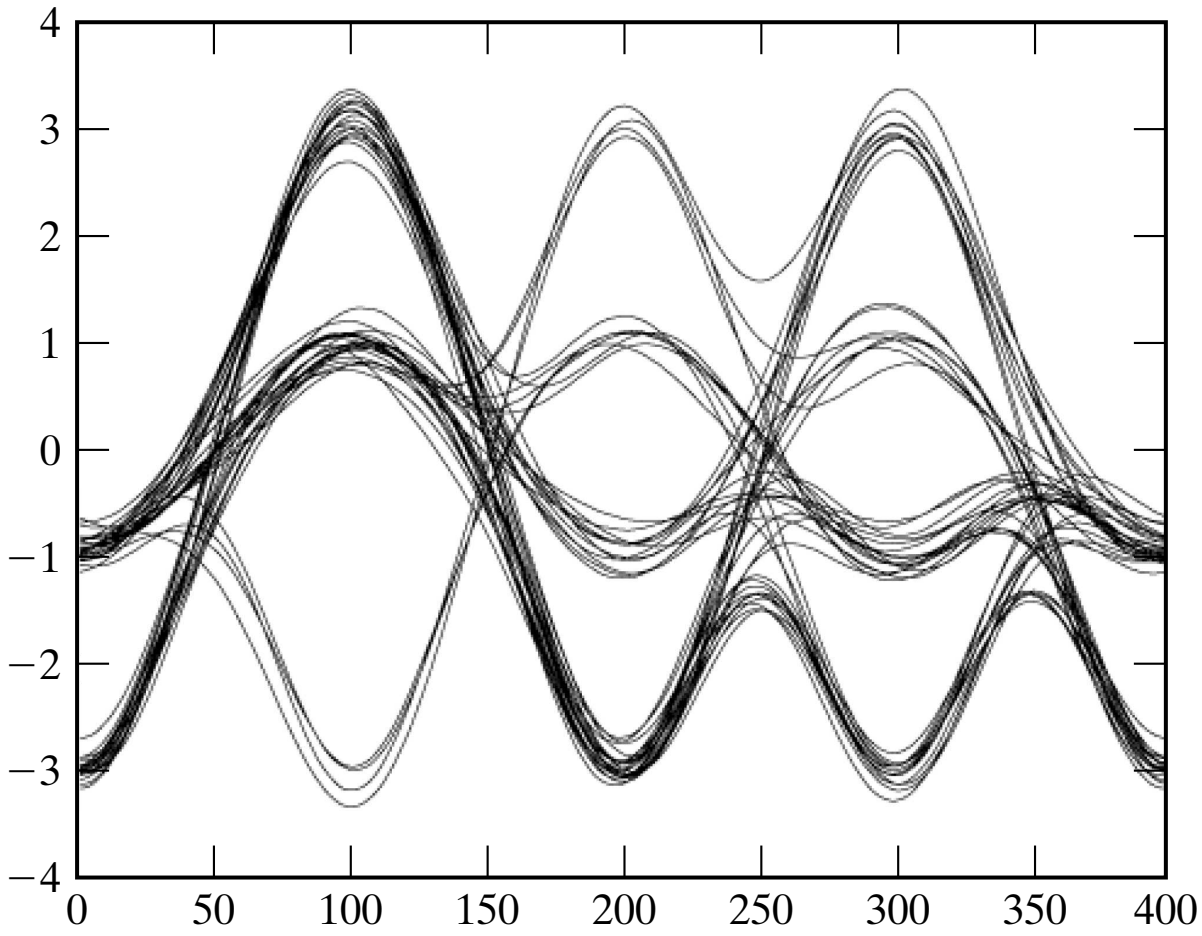
What if ... (cont'd)

Channel noise (cont'd): Received signal eye diagram of 4 symbol wide overlays



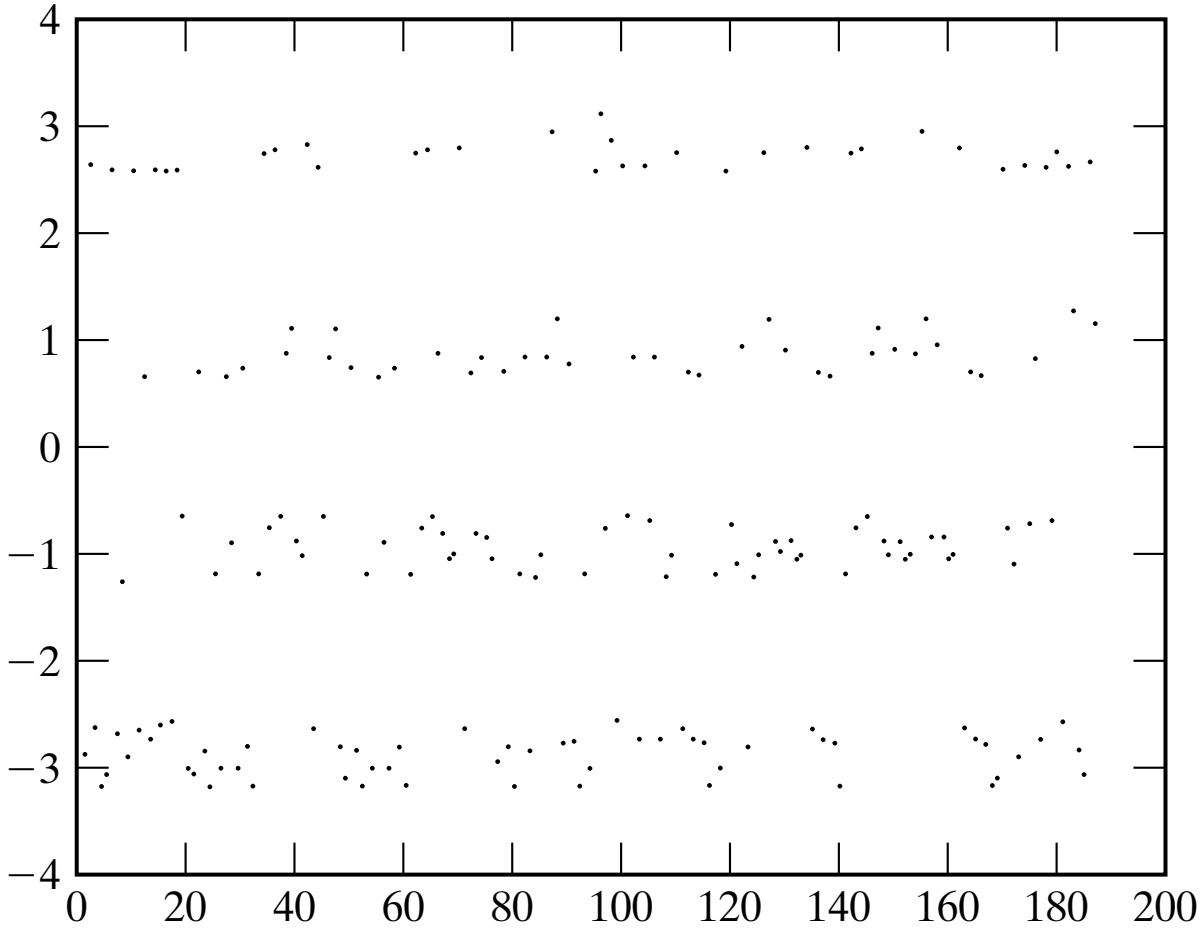
What if ... (cont'd)

Channel noise (cont'd): Pulse correlator filter synchronized output signal



What if ... (cont'd)

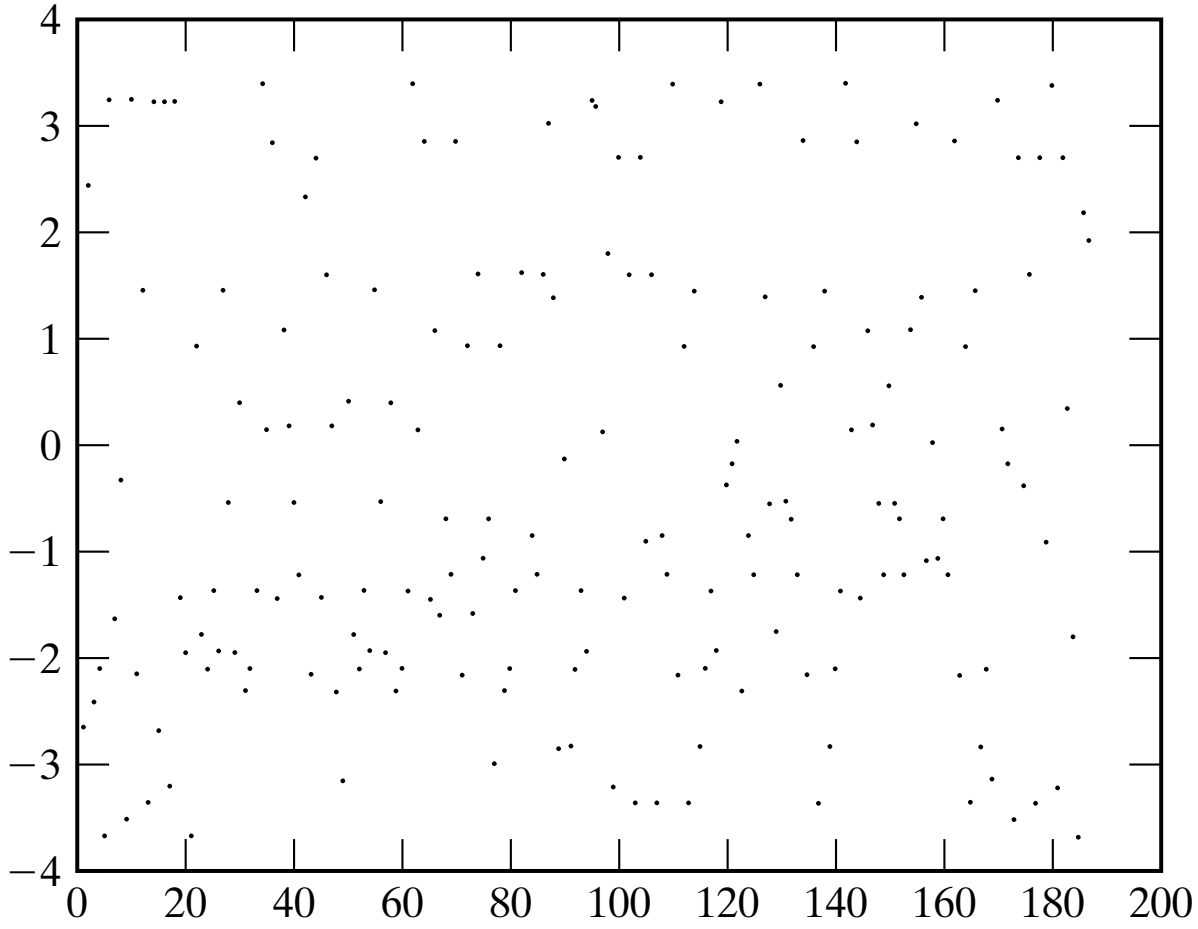
Multipath: Mild multipath soft decisions



The appearance of 4 distinct stripes indicates no decision errors.

What if ... (cont'd)

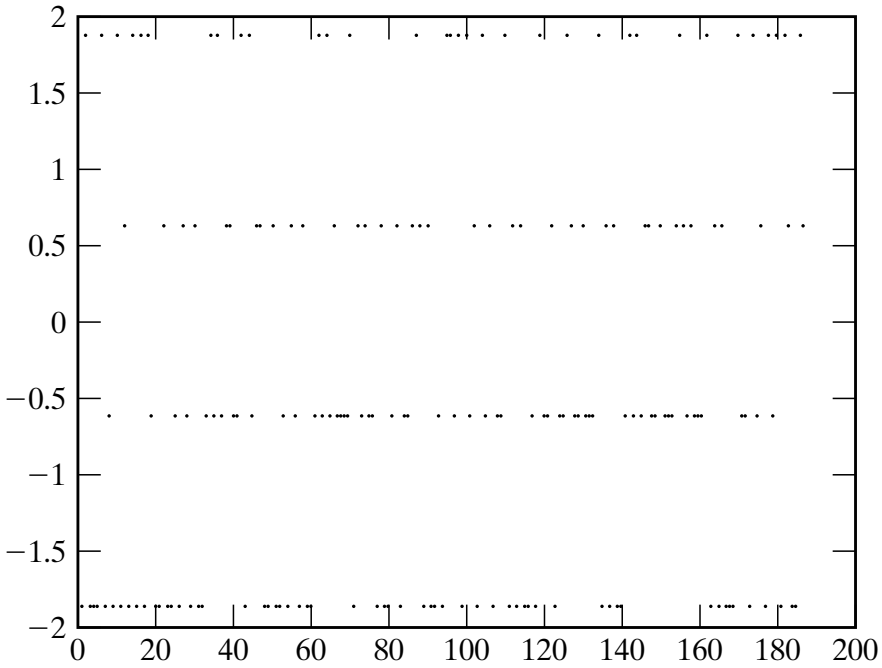
Multipath (cont'd): Harsh multipath soft decisions



The lack of emergence of 4 distinct stripes indicates the (likely) presence of decision errors.

What if ... (cont'd)

Carrier phase offset: Severe offset

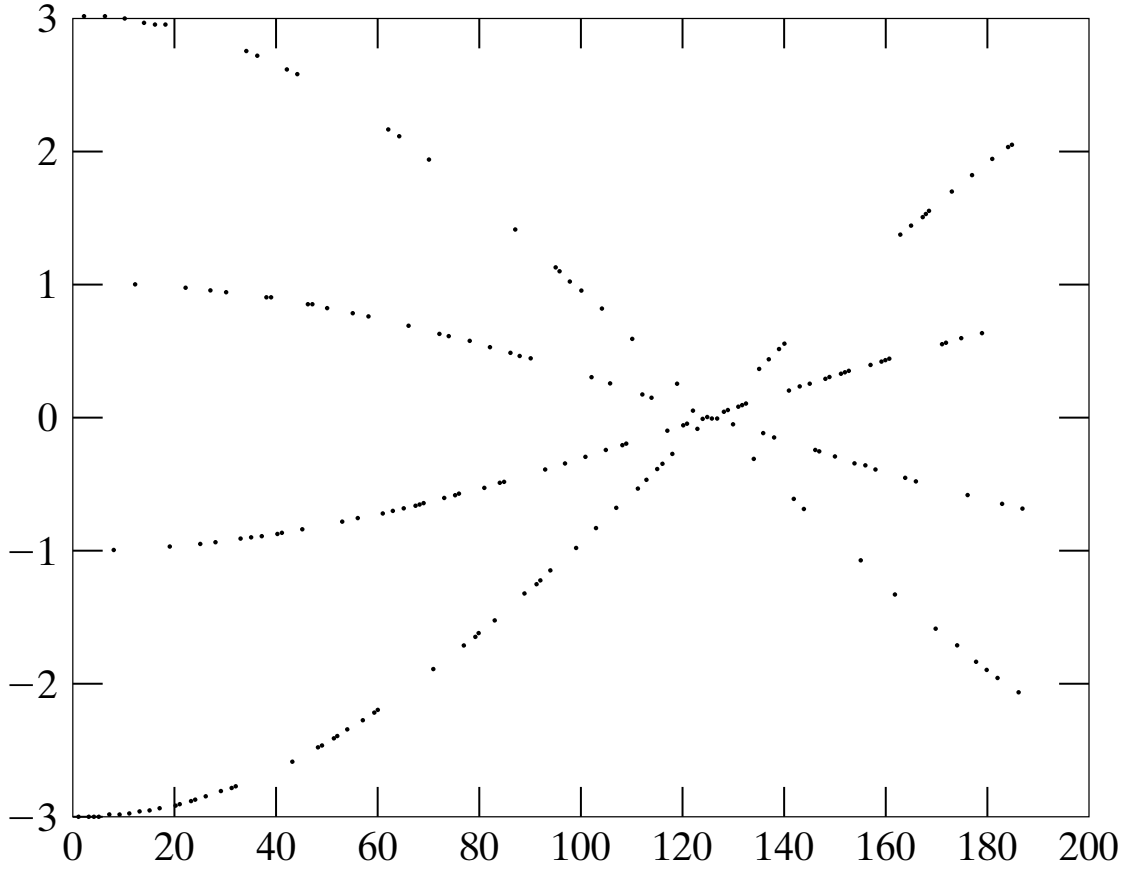


The attenuation due to carrier phase offset reduces all soft decisions below magnitude 2 resulting in no ± 3 as decision device outputs $\Rightarrow$ plenty of errors.

If scaled back up so stripes of largest magnitude values are above magnitude 2, the SNR will suffer relative to case without carrier phase offset.

What if ... (cont'd)

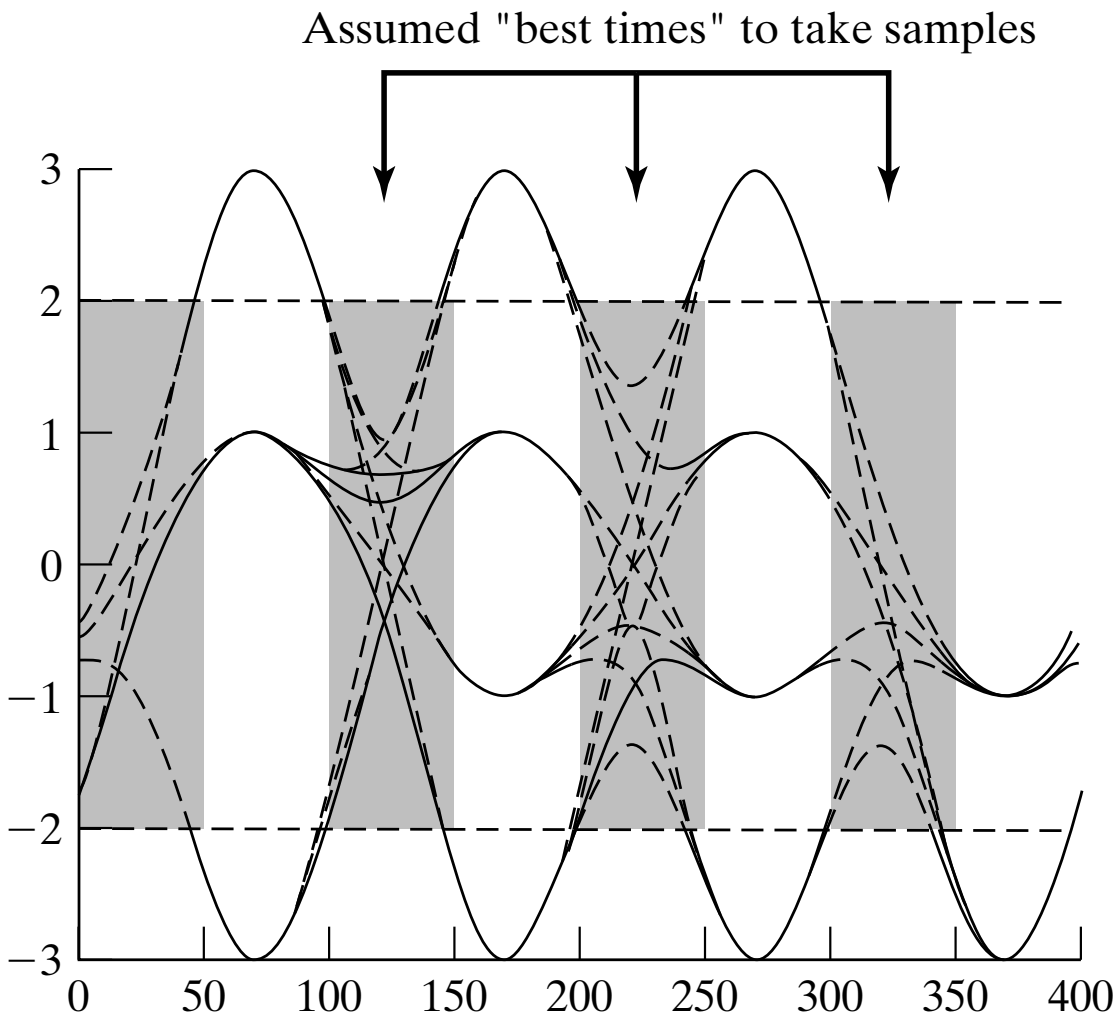
Carrier frequency offset: Soft decisions for 0.01% frequency offset



The carrier frequency offset appears as a low frequency amplitude modulation of the desired outputs.

What if ... (cont'd)

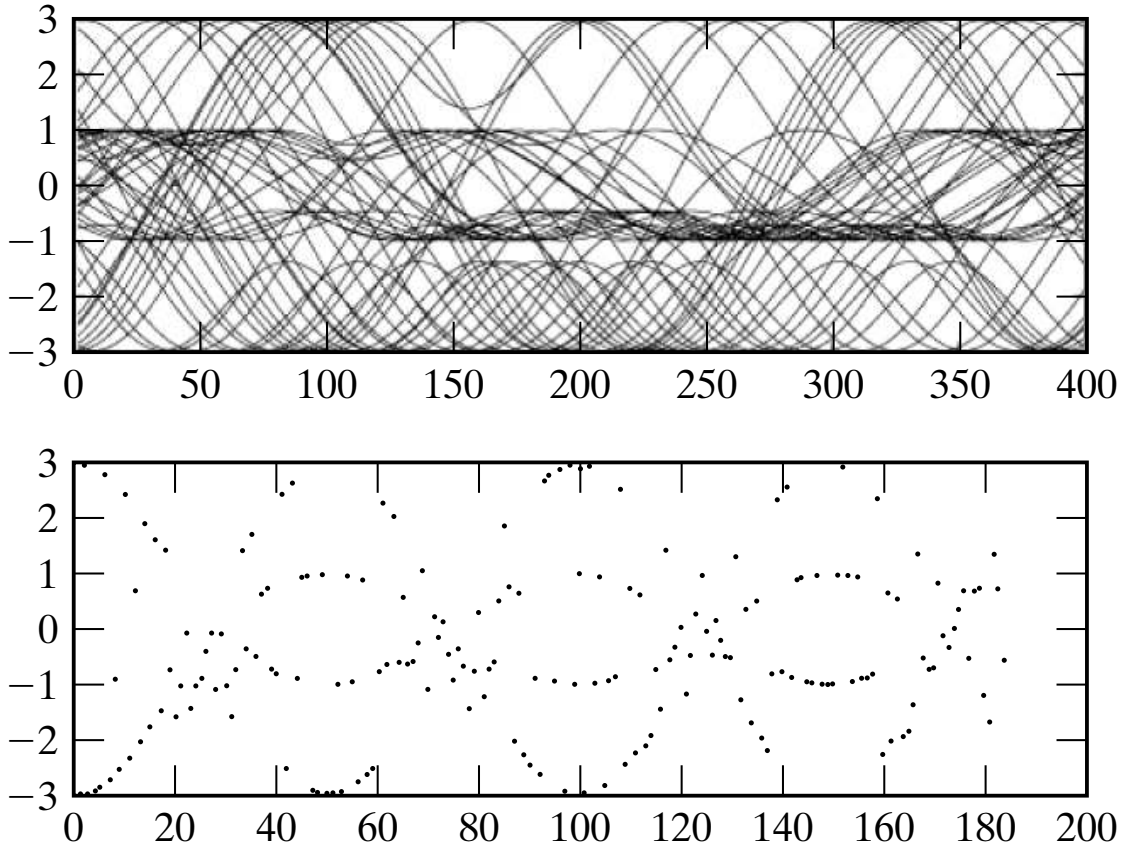
Downsampler timing offset: Eye diagram with debilitating offset



With samples for symbol values taken every 100 samples after sample 125, numerous errors occur.

What if ... (cont'd)

Downsampler period offset: Eye diagram (top) and soft decisions (bottom) with 1% downsampler period offset

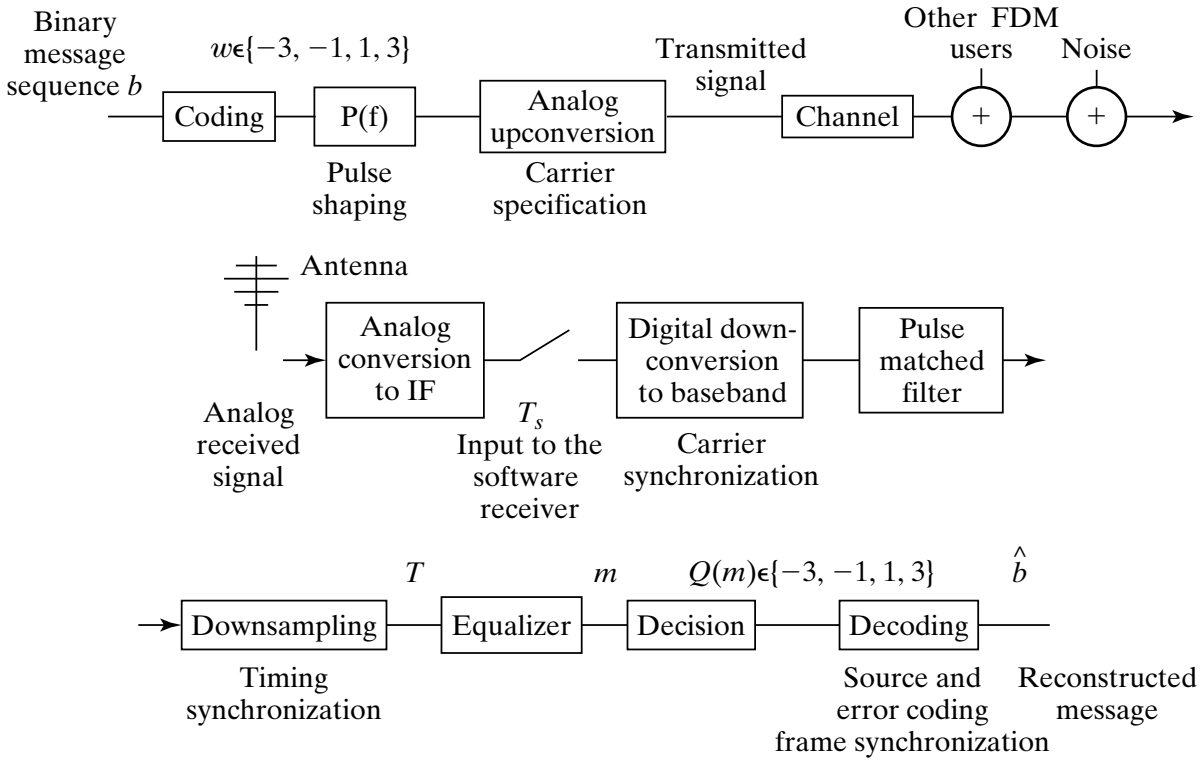


All is lost...

Well, then...

- Coding and matched receive filtering are intended to counter effects of broadband channel noise.
- Equalization compensates for multipath interference, and can reject narrowband interferers as well.
- Carrier recovery schemes (including phase locked loops and Costas loops) adjust receiver oscillator phase to counteract phase offset (and mild frequency offset).
- Timing recovery (using interpolation) is intended for reduction of downsampler timing offset (and mild period offset).

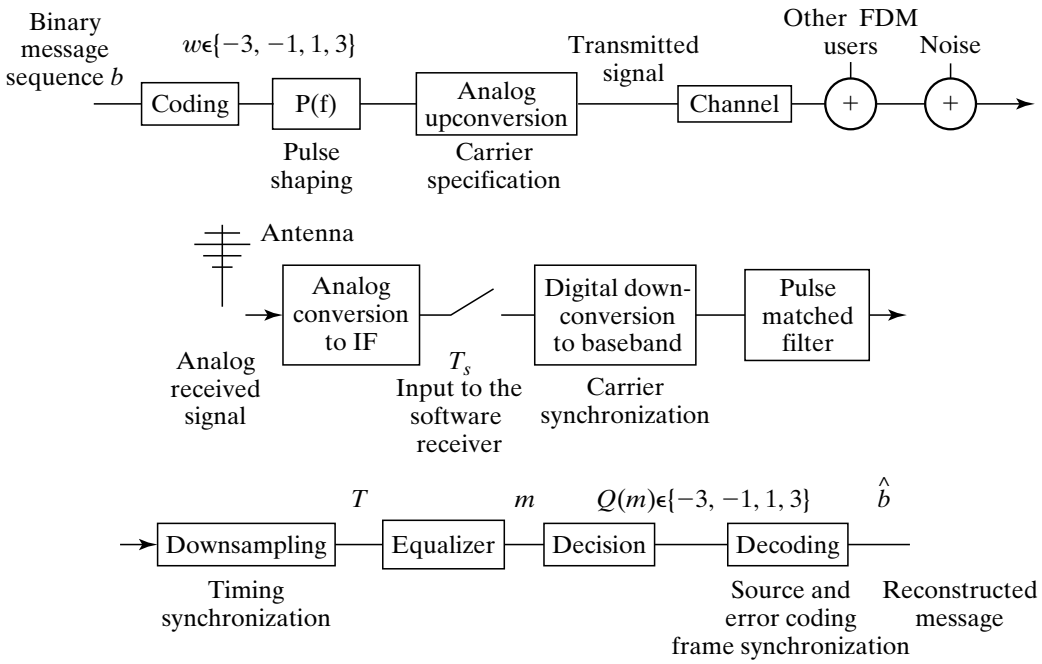
Our Project System



CARRIER RECOVERY

- ⋆ Carrier Phase Tracking
- ⋆ Adaptive Algorithm Development
- ⋆ Carrier Extraction
- ⋆ Phase-locked Loop
- ⋆ Costas Loop

Carrier Phase Tracking



- A fixed phase offset between the transmitter and carrier oscillators results in an attenuation in the downconverted signal by the cosine of this phase difference.
- We seek algorithms for adjusting the receiver mixer's phase that can track (slow) time variations in the transmitter's phase.
- We treat carrier phase tracking as a single-parameter adaptation problem.

Adaptive Algorithm Development

Our (single-parameter) adaptive algorithm development strategy:

- Propose a cost function assessing behavior over measured data set.
- Check location of minima and maxima in terms of adjusted parameter to see if in desired location.
- Pursue (small stepsize) gradient descent strategy (with its commutability of averaging and differentiation). The correction term must be calculable from available signals.
- Test performance.

Carrier Extraction

- For AM with suppressed carrier we will process the received upconverted signal

$$r(kT_s) = s(kT_s)\cos(2\pi f_0 kT_s + \phi)$$

  which does not include an additive carrier, in order to extract a signal related to the carrier.

- Consider squaring the received signal and using $\cos^2(x) = (1/2)(1 + \cos(2x))$ to produce $r^2(kT_s) =$

$$(1/2)s^2(kT_s)[1 + \cos(4\pi f_0 kT_s + 2\phi)]$$

Carrier extraction (cont'd)

- Rewrite $s^2(t)$ as the sum of its (positive) average value and the variation about this average $s^2(kT_s) = s^2_{avg} + v(kT_s)$, so

$$r^2(kT_s) = \frac{1}{2}s^2(kT_s)[1 + \cos(4\pi f_0 kT_s + 2\phi)]$$

$$= (1/2)[s^2_{avg} + v(kT_s) + s^2_{avg}\cos(4\pi f_0 kT_s + 2\phi)$$

$$+v(kT_s)\cos(4\pi f_0 kT_s + 2\phi)]$$

- A narrow bandpass filter centered at $2f_0$ with phase shift ρ at $2f_0$ extracts

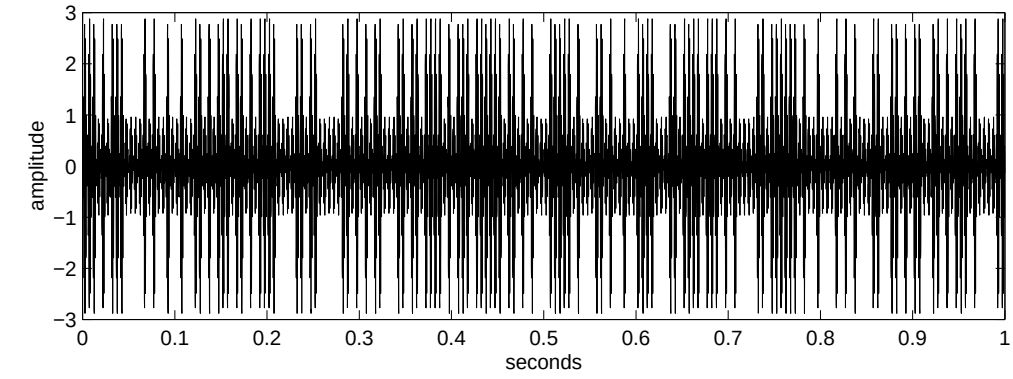
$$x(kT_s) = (1/2)s^2_{avg}\cos(4\pi f_0 kT_s + 2\phi + \rho)$$

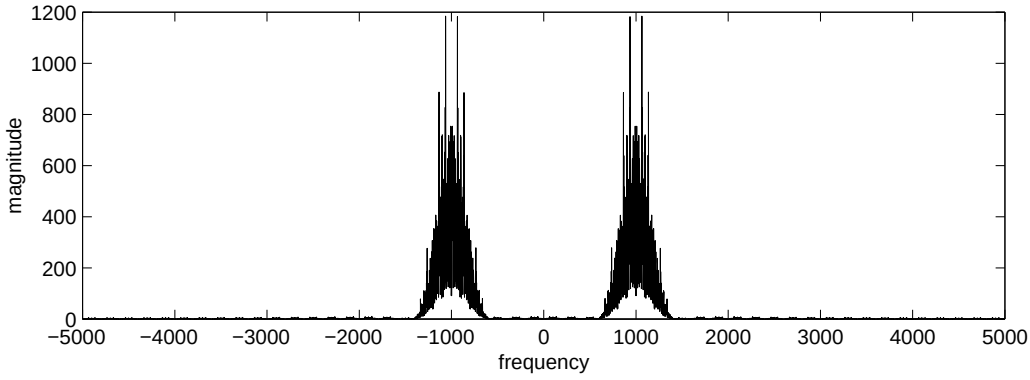
  from r^2 while passing a bit of v about $2f_0$.

- Digital BPF implementation presumes that $2f_0$ lies within the Nyquist frequency $1/(2T_s)$.

Carrier Extraction (cont'd)

For 1 second of a 4-PAM signal with Hamming blip symbol width $T = 0.005$, sample period (with an oversample factor of 50) $T_s = 0.0001$, and a carrier with frequency $f_0 = 1000$ and phase $\phi = -1$, (from `pllcrt`) the received signal and its spectrum are



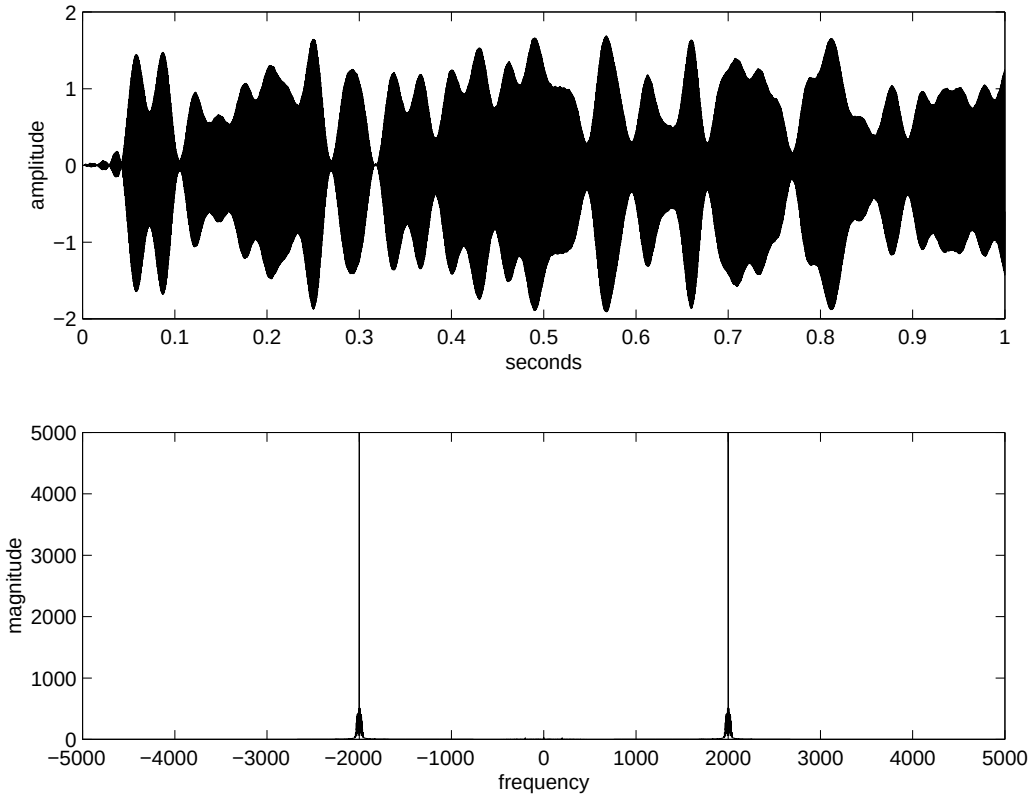


Carrier Extraction (cont'd)

Passing the received signal with $f_0 = 1000$

$$r(kT_s) = s(kT_s)\cos(2\pi f_0 kT_s + \phi)$$

through a squarer and a BPF centered at 2000 Hz with approximately 100 Hz passband and $\text{mod}(\rho, 2\pi)$=0 (where $\text{mod}(a, b)$ produces the remainder after division of a by b) yields x in time and frequency



Phase-locked Loop (PLL)

To introduce a phase-locked loop, the most widely known carrier recovery scheme, we present a candidate cost function producing the PLL.

- Reconsider the output of the squarer and narrow BPF, which is a scaled version of the carrier $x(kT_s) = g\cos(4\pi f_0 kT_s + 2\phi)$ where g is $s_{avg}^2/2$ times the square of the product of the channel and BPF gains at $2f_0$ and ψ is the BPF phase (mod 2π) at $2f_0$.
- Consider downconverting $x(kT_s)$ with our (unsynchronized) receiver oscillator's output and form

$$x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)$$

$$\approx g\cos(4\pi f_0 kT_s + 2\phi + \psi)\cos(4\pi f_0 kT_s + 2\theta + \psi)$$

$$= \frac{g}{2}\{\cos(2\phi - 2\theta) + \cos(8\pi f_0 kT_s + 2\phi + 2\theta + 2\psi)\}$$

PLL (cont'd)

- Lowpass filtering this product with a LPF with cutoff below $4f_0$ produces

$$\text{LPF}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}$$

$$\approx \frac{g}{2}\cos(2\phi - 2\theta)$$

  which is maximized when $2\phi - 2\theta = 2n\pi \Rightarrow \phi - \theta = n\pi$.

- Value of positive, finite g does not effect locations of maxima and minima.

- We will choose to maximize

$$J_{PLL} = \frac{1}{P}\sum_{k=k_0}^{k_0+P}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}$$

$$= \text{avg}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}$$

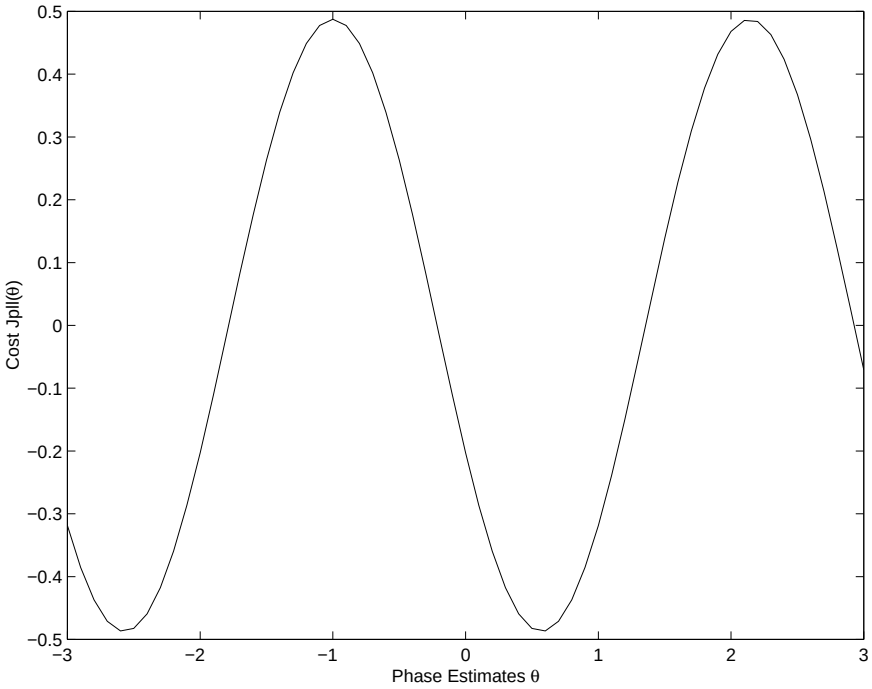
$$\sim \text{LPF}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}$$

PLL (cont'd)

As a numerical test for extrema, the PLL cost

$$J_{PLL} = \text{LPF}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}$$

can be formed for various fixed θ producing (via `pllcrt`)



A maximum (near 0.5 with $g \approx 1$ in this case) appears at the desired location of $\theta = \phi = -1$ (with $\psi = 0$) and at locations an integer multiple of π away, as predicted in the preceding analysis.

PLL (cont'd)

Following a gradient ascent strategy for maximization, compose

$$\theta[k+1] = \theta[k]$$

$$+\bar{\mu}\frac{\partial}{\partial\theta}[\text{avg}\{x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)\}]|_{\theta=\theta[k]}$$

With a small stepsize assuring (approximate) commutability of differentiation and average

$$\theta[k+1] = \theta[k]$$

$$+\bar{\mu}\cdot\text{avg}\{\frac{\partial}{\partial\theta}[x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)]|_{\theta=\theta[k]}\}$$

where

$$\frac{\partial}{\partial\theta}[x(kT_s)\cos(4\pi f_0 kT_s + 2\theta + \psi)]|_{\theta=\theta[k]}$$

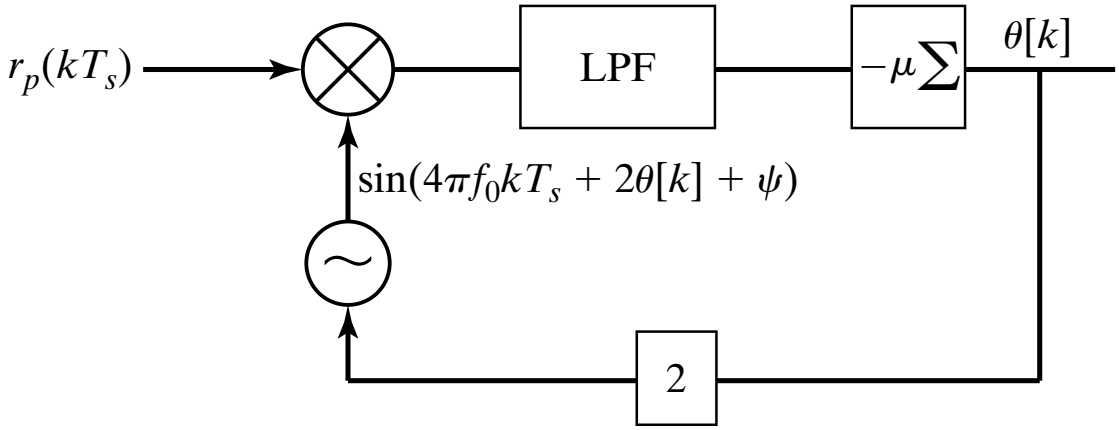
$$= -2x(kT_s)\sin(4\pi f_0 kT_s + 2\theta[k] + \psi)$$

This produces

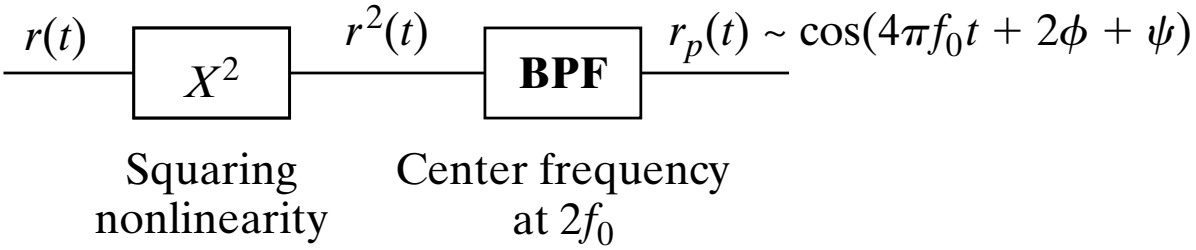
$$\theta[k+1] = \theta[k] - \mu\text{LPF}\{x(kT_s)\sin(4\pi f_0 kT_s + 2\theta[k] + \psi)\}$$

PLL (cont'd)

PLL carrier recovery system:



where input r_p is the processed received signal of



and "normalizing" gain $(2/s_{avg}^2)$ has been implicitly included in BPF (though any substantial gain is acceptable) which has phase shift ψ at frequency $2f_0$.

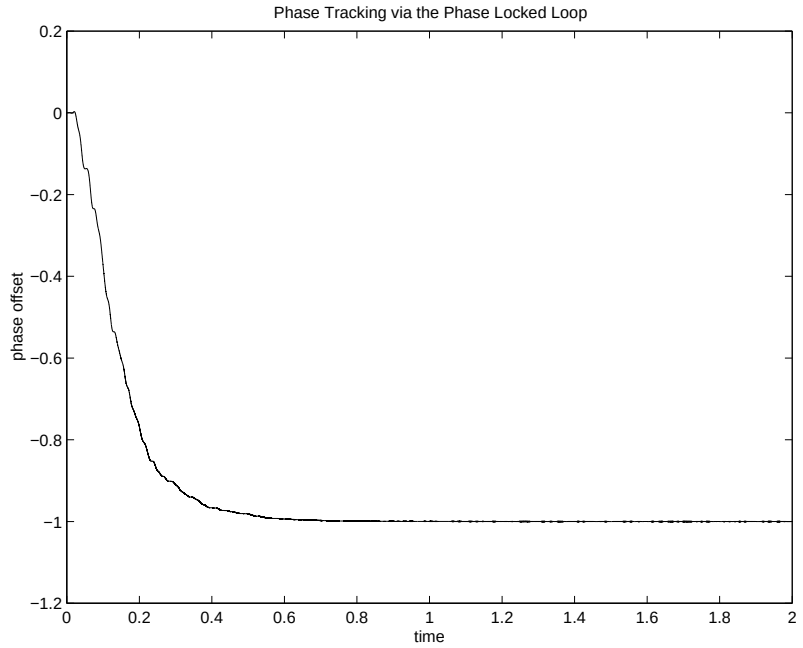
When ψ is nonzero, it should be added in carrier recovery system schematic after $2\theta[k]$ term in the oscillator.

PLL (cont'd)

For the PLL algorithm with explicit LPF preceding integrator/summer removed

$$\theta[k+1] = \theta[k] - \mu x(kT_s)\sin(4\pi f_0 kT_s + 2\theta[k] + \psi)$$

a typical learning curve (from `pllcrt`) for a stepsize of $\mu = 0.001$ for our continuing example (with $\psi = 0$ and an objective of $\theta = -1$) is



Costas Loop

Now, we seek an algorithm not based on a presumption of carrier extraction from the received signal.

- Reconsider the received signal

$$r(kT_s) = s(kT_s)\cos(2\pi f_0 kT_s + \phi)$$

and form

$$2r(kT_s)\cos(2\pi f_0 kT_s + \theta)$$

$$= s(kT_s)[\cos(\phi - \theta) + \cos(4\pi f_0 kT_s + \phi + \theta)]$$

- With a LPF cutoff below $2f_0$

$$\text{LPF}\{2r(kT_s)\cos(2\pi f_0 kT_s + \theta)\}$$

$$= v(kT_s)\cos(\phi - \theta)$$

where $v(kT_s) = \text{LPF}\{s(kT_s)\}$. If the cutoff frequency of the LPF is above the bandwidth of the baseband waveform s, then v is s.

Costas Loop (cont'd)

- As a cost function, consider

$$\frac{1}{P}\sum_{k=k_0-(P-1)}^{k_0}\{\text{LPF}[2r(kT_s)\cos(2\pi f_0 kT_s+\theta)]\}^2$$

$$\approx \text{avg}\{v^2(kT_s)\cos^2(\phi-\theta)\}$$

- Because the squared cosine term is fixed,

$$\text{avg}\{v^2(kT_s)\cos^2(\phi-\theta)\}$$

$$=\left(\text{avg}\{v^2(kT_s)\}\right)\frac{(1+\cos(2(\phi-\theta)))}{2}$$

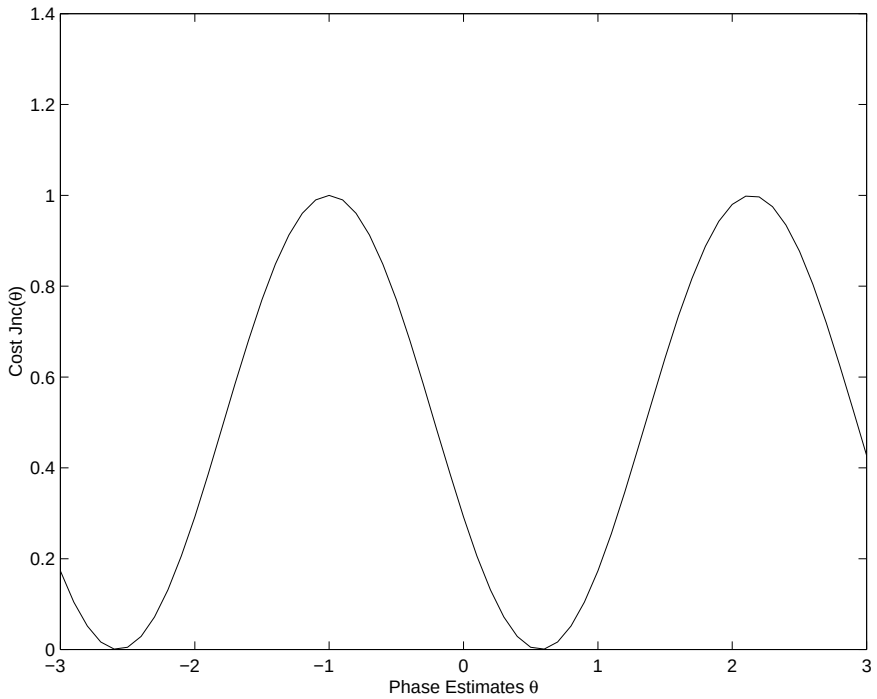
and assuming that the average of v^2 is fixed, this cost function will be maximized with a value equal to the average of v^2 (which is average value of $\{\text{LPF}[\text{s}]\}^2$) at $\phi-\theta=\pi n$ or $\theta=\phi+\pi n$ for all (positive and negative) integers n.

Costas Loop (cont'd)

We can numerically check the extrema of a normalized cost

$$J_{NC} = \frac{\frac{1}{P}\sum_{k=1}^{P}\left(\text{LPF}\{2r(kT_s)\cos(2\pi f_0 kT_s + \theta)\}\right)^2}{\frac{1}{P}\sum_{k=1}^{P}\left(\text{LPF}\{s(kT_s)\}\right)^2}$$

where r is the received signal for our continuing example for various fixed θ producing (via `ccrt`)



This normalized cost function matches $(1 + \cos(2(\phi - \theta)))/2$, as anticipated.

Costas Loop (cont'd)

Our next step in our algorithm creation strategy is to interchange the averaging and differentiation in the gradient ascent update

$$\theta[k+1] = \theta[k] + \bar{\mu}\frac{\partial}{\partial\theta}[\text{avg}\{(\text{LPF}\{2r(kT_s)$$

$$\cdot\cos(2\pi f_0 kT_s + \theta)\})^2\}]|_{\theta=\theta[k]}$$

With

$$\text{LPF}\{2r(kT_s)\cos(2\pi f_0 kT_s + \theta)\} = v(kT_s)\cos(\phi - \theta)$$

the update can be written as

$$\theta[k+1] = \theta[k] + \bar{\mu}\cdot\text{avg}\{\frac{\partial}{\partial\theta}[v^2(kT_s)\cos^2(\phi-\theta)]|_{\theta=\theta[k]}\}$$

$$= \theta[k] + \mu\cdot\text{avg}\{v^2(kT_s)(\cos(\phi-\theta)\frac{\partial\cos(\phi-\theta)}{\partial\theta})|_{\theta=\theta[k]}\}$$

and from $\frac{d}{dx}(\cos(y)) = -(\sin(y))\frac{dy}{dx}$ we wish to form

$$\theta[k+1] = \theta[k]$$

$$+\mu\cdot\text{avg}\{v^2(kT_s)\cos(\phi-\theta[k])\sin(\phi-\theta[k])\}$$

Costas Loop (cont'd)

Given

$$\text{LPF}\{2r(kT_s)\cos(2\pi f_0 kT_s+\theta)\} = v(kT_s)\cos(\phi-\theta)$$

to compose the update from measurable signals we need to find a realizable expression for $v(kT_s)\sin(\phi-\theta)$.

For a LPF with cutoff under $2f_0$, defining $v = \text{LPF}\{s\}$ and using $\sin(x)\cos(y) = (1/2)[\sin(x-y)+\sin(x+y)]$ and $\sin(-x) = -\sin(x)$ produces

$$\text{LPF}\{2r(kT_s)\sin(2\pi f_0 kT_s+\theta)\}$$

$$= \text{LPF}\{s(kT_s)\cos(2\pi f_0 kT_s+\phi)\sin(2\pi f_0 kT_s+\theta)\}$$

$$= \text{LPF}\{s(kT_s)(\sin(\theta-\phi)-\sin(4\pi f_0 kT_s+\phi+\theta))\}$$

$$= -v(kT_s)\sin(\phi-\theta)$$

Costas Loop (cont'd)

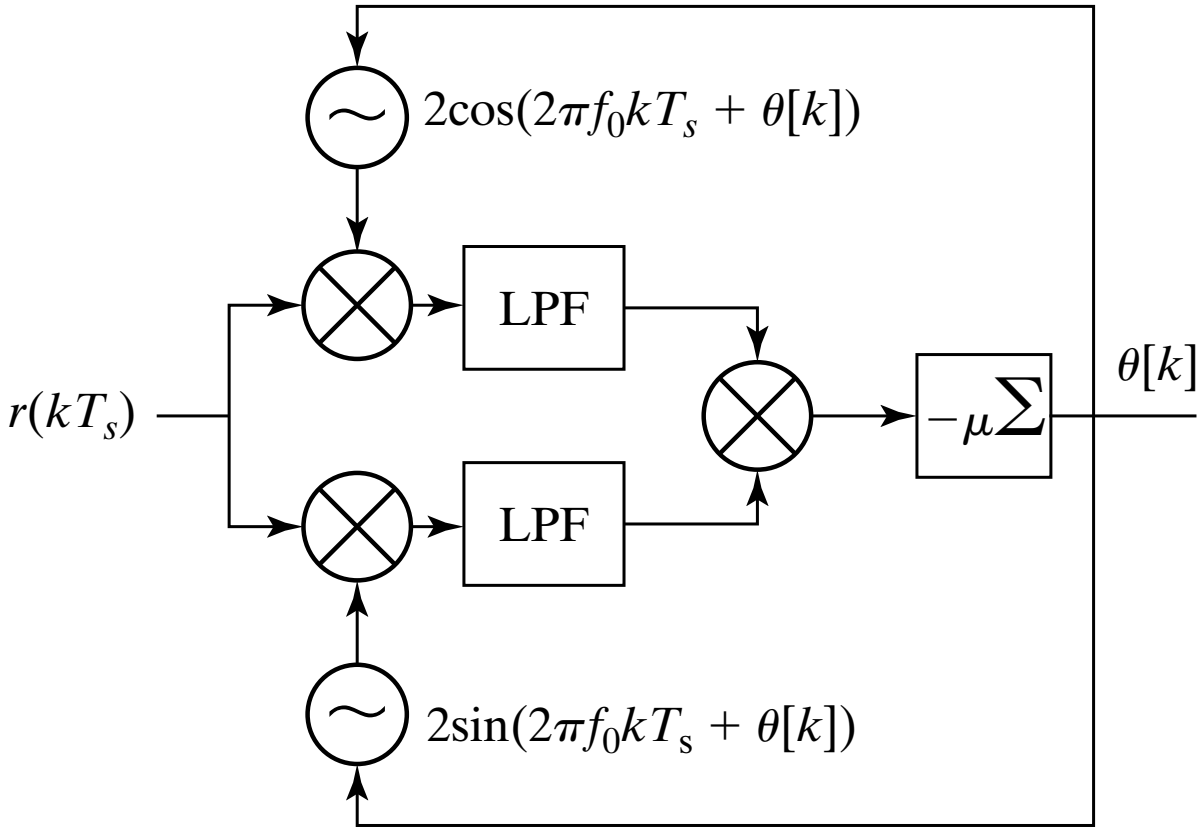
Thus, a small stepsize gradient ascent algorithm (for maximization of J_C) is

$$\theta[k+1] = \theta[k] - \mu \cdot \text{avg}[\text{LPF}\{2r(kT_s)\cos(2\pi f_0 kT_s + \theta[k])\} \cdot \text{LPF}\{2r(kT_s)\sin(2\pi f_0 kT_s + \theta[k])\}]$$

- The use of lowpass filtering in the update is predicated on a presumption that the LPF output is characterized by its asymptotic response.
- This effectively presumes $\theta[k]$ remains fixed for a sufficiently long time for this asymptotic behavior to be achieved.
- We rely on a small stepsize μ to keep $\theta[k]$ variations modest in the (relatively) short time frame anticipated for LPF achievement of asymptotic behavior.

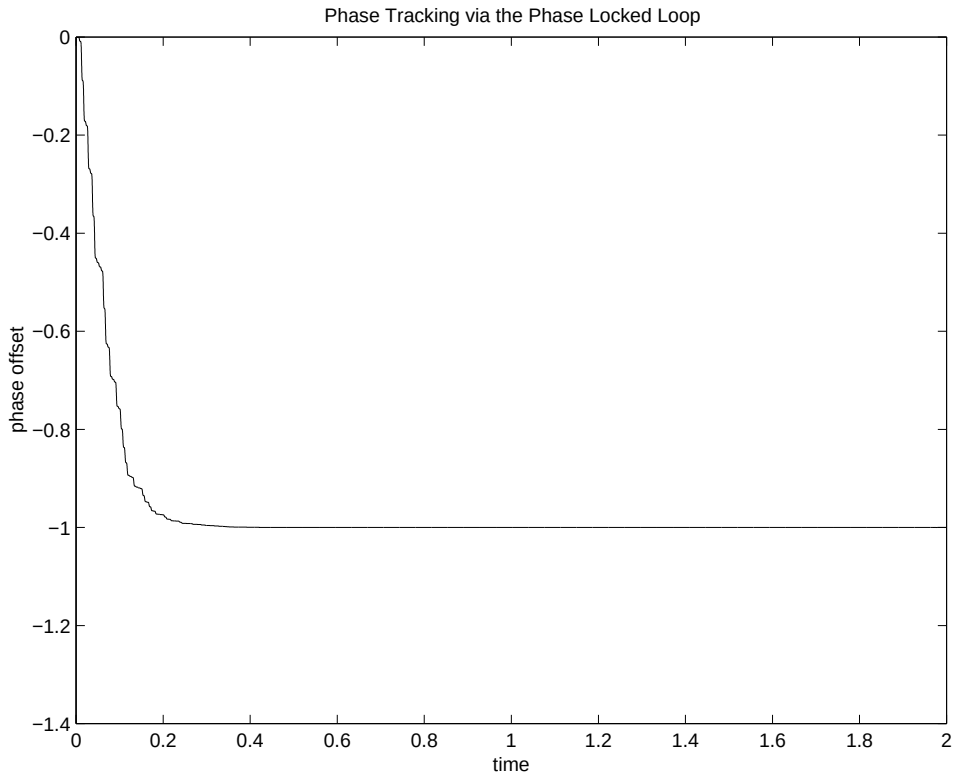
Costas Loop (cont'd)

Schematic for Costas loop carrier phase recovery with the "outer" averaging removed (which presumes that the integrator/summer of the update will provide sufficient averaging):



Costas Loop (cont'd)

A typical learning curve for this Costas loop carrier phase recovery scheme (as shown in the preceding schematic without explicit averaging in the update) on our continuing example (with an objective of -1) is (from `ccrt` with a stepsize of $\mu = 0.001$)



Laboratory Exercises – Day 2

Introduction to Digital Communication Receiver Design

Task 1: Understanding the Subsampled-IF Receiver Architecture

In an IF receiver (also called a *heterodyne* receiver), the downconversion from RF is done in 2 steps:

- An analog circuit downconverts to some intermediate frequency, where the signal is sampled.
- The resulting signal is then digitally downconverted to baseband.

The advantage of this 2-step method is that the analog downconversion can be performed with minimal precision (and hence inexpensively), while the sampling can be done at a reasonable rate.

In a standard sampled-IF receiver, the sampling frequency is typically chosen to be twice the IF frequency (i.e. the Nyquist rate). However, another class of IF receivers called *subsampled IF receiver* uses a sampling frequency lower than the Nyquist rate, which results in aliasing. However, the aliasing is introduced in a way that reconstruction of the signal is still possible. Recall that sampling introduces copies of the signal at every multiple of the sampling rate.

To illustrate the subsampled IF receiver architecture, we consider an specific example with the following parameters:

parameter	value
carrier frequency	$f_{RF} = 1$ GHz
intermediate frequency	$f_{IF} = 2$ MHz
receiver sampling rate	$f_s = 850$ kHz
signal bandwidth	$B = 100$ kHz

where the signal bandwidth of the baseband signal is defined as having spectral content between $-B$ and $+B$.

Your task: Draw the spectrum of the signal at each of the following steps

1. The original baseband signal with bandwidth B.
2. The signal after modulation to the RF frequency, accomplished by mixing with a sinusoid of frequency f_{RF}.
3. The signal after downconversion to the IF frequency, accomplished by mixing with a sinusoid of frequency $f_{RF} - f_{IF}$.
4. The signal after bandpass filtering, which removes the unwanted "image".

5. The signal after (sub)sampling at rate f_s (Note: For this one, you only need to draw the spectrum between $-f_s/2$ and $+f_s/2$).

If the next step were to perform downconversion of the signal to baseband, what frequency would you choose for the sinusoid used in the downconversion?

In spite of the fact that subsampling introduces aliasing, is it still possible to recover the original baseband signal? Or is the signal distorted?

A subsampled-IF receiver is attractive because it can be implemented even more inexpensively than a standard sampled-IF receiver. However, there is one major drawback to this use of this receiver architecture in the presence of noise (AWGN). Can you think of what this drawback might be?

Task 2: Implementing the Costas Loop

The receiver you have been given currently uses a PLL for carrier recovery (in `/system_code/Rx.m`). Your task is to replace the PLL with a Costas loop, and compare the performance of the two schemes.

Recall from the lecture notes, the update equation for the Costas loop has the form

$$\theta[k+1] \;=\; \theta[k] - \mu \cdot \text{LPF}\left\{2r(kT_s)\cos(2\pi f_0 kT_s + \theta[k])\right\} \cdot \text{LPF}\left\{2r(kT_s)\sin(2\pi f_0 kT_s + \theta[k])\right\}$$

Since the existing receiver code has a PLL, it is useful to compare and contrast the two algorithms in terms of their implementation. While the PLL requires a pre-processing step, the Costas loop does not require pre-processing. The Costas loop makes use of a low pass filter, which is not present in the PLL, and you will need to use `remez` to design this filter. The schematic for the Costas loop on the next to last page of the lecture notes for DAY 2 may be helpful, as well.

In testing your Costas loop implementation, you should start by using the most benign conditions (no noise, no channel, no phase noise). Once your implementation is working, you should gradually add more realistic channel impairments. Additionally, you will need to tune the Costas loop parameters (stepsize, filter parameters, etc) for best performance.

In the following exercises, you should check how the receiver performs in comparison to the old PLL-based receiver. You should always include a plot of the phase estimate, θ.

1. Modify the channel, decrease the SNR, or increase the phase noise variance. In general, do you find one receiver to be more robust?

2. Is one receiver better at tracking the variation due to phase noise? Explain this by referring to the plot of θ.

3. What effect to you observe as you decrease the stepsize? What happens as you increase the stepsize?

Transmitter/Receiver Code Description

Introduction to Digital Communication Receiver Design

Introduction

A transmitter and sampled-IF receiver have been implemented in MATLAB, and this document describes the corresponding code. This operation of the receiver, including its chosen parameters, are described in the latter half of the lecture notes for DAY 4 under the heading of "Putting It All Together: Receiver Design". This receiver system will be used in the lab assignments for Days 2-4 by focusing only on the specific segment described in the associated lectures, while allowing us to judge the impact on overall system performance. The block diagram in Fig. 1 shows the steps of generation of the transmitted signal, its propagation through the channel, and the operations performed by the receiver. While

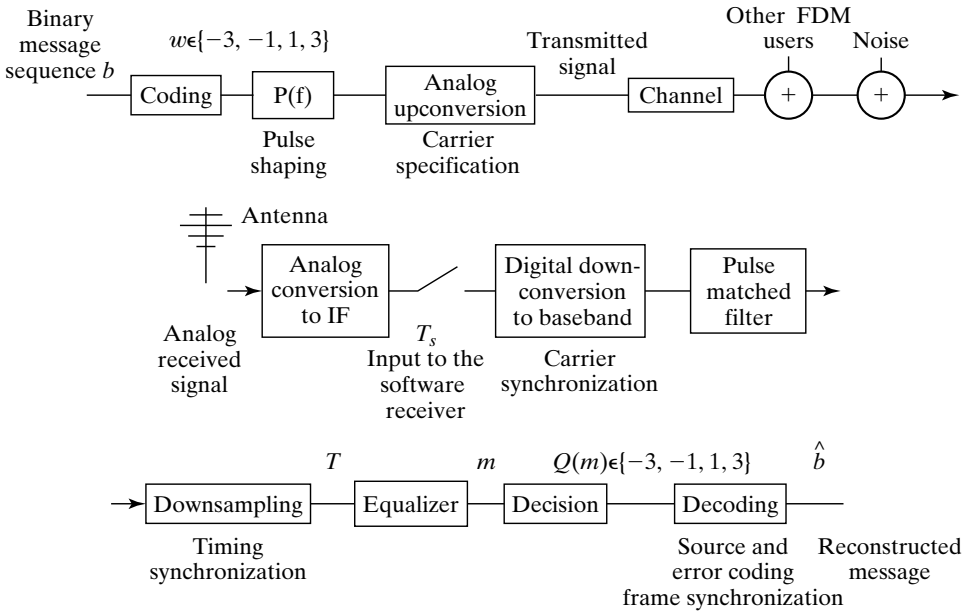


Figure 1: System Block Diagram

the blocks in this figure are quite general, design choices were made in the development of this particular transmitter/receiver implementation. These design choices (i.e. which algorithms have been selected) and the code to implement them will be described in the following sections.

MATLAB Files

A brief description of each of the functions used in the complete system is provided here. You can find all of the files in the `/system_code` directory. For a more detailed description

of the inputs and outputs for each of these functions, you can using the `help` command at the MATLAB prompt (e.g. by typing "`help Tx`").

Main Files

Listings at the end of this code description document.

- `main.m` — This script is merely an example which shows how to set up the system parameters, run the transmitter, run the receiver, and calculate the bit-error-rate.
- `Tx.m` — This function contains the transmitter, and introduces the impairments (e.g. the channel, imperfect receiver frontend, etc.)
- `Rx.m` — This function decodes the received signal, and outputs the message.
- `globalParams.m` — This function contains the parameters for the system (e.g. sampling period, IF frequency, marker sequence, etc.)

Subroutine Files

These files are from *Telecommunication Breakdown*.

- `letters2pam.m` — This function converts an ASCII text sequence into 4-PAM symbols. Used by `Tx.m`.
- `pam2letters.m` — This function converts a sequence of 4-PAM symbols into an ASCII text string. Used by `Rx.m`.
- `quantalph.m` — This function is effectively a minimum Euclidean distance detector, or decision device. It accepts "soft" PAM symbols, and quantizes the input to the nearest PAM symbol. Used by `Rx.m`.
- `srrc.m` — This function generates the impulse response for the square-root raised-cosing pulse shape. Used by both `Tx.m` and `Rx.m`.
- `interpsinc.m` — The function performs sinc interpolation, and is used for the baud-timing and downsampling in the receiver. Used by `Rx.m`.

Transmitter Details (`Tx.m`)

This section briefly describes each of the main components of the transmitter, and points to their corresponding line numbers in the code. The components can also be found in the block diagram in Fig. 1. Note that, in addition to the transmitter, `Tx.m` also includes the channel and receiver frontend blocks.

- **Calculate Intermediate Variables** (lines 19-29) — This part calculates intermediate variables which are used in the transmitter, including the upsampling/downsampling ratios and the phase noise random process.
- **Generation of 4-PAM sequence** (lines 30-32) — This part encodes the ASCII test message into 4-PAM signals (using `letters2pam`), and inserts the header and marker sequences. The result is a serial stream of 4-PAM symbols stored in the variable called `s`.
- **Pulse Shaping** (lines 33-35) — This part performs upsampling of the 4-PAM signal and then filters the signal with the pulse shape obtained from `srrc`. The result is stored in the variable `x`.
- **Analog Upconversion** (lines 36-39) — This part modulates the signal up to the carrier frequency, and includes the effect of phase noise. The result is stored in `x_rf`.
- **Channel** (lines 40-42) — This part convolves the upconverted signal with the channel, storing the result in the variable `x2`.
- **Noise** (lines 43-47) — This part adds the additive white Gaussian noise of specified SNR.
- **Analog Conversion from RF to IF** (lines 48-52) — This part acts as the frontend of the receiver, and performs analog conversion of the RF signal down to IF. While the RF signal in reality would be analog, our computer simulation uses a digital representation throughout; thus, the sampled-IF receiver is obtained from the RF signal by simple downsampling. The result is the digital signal which gets passed into `Rx.m`.

Receiver Details (`Rx.m`)

Similar to the previous section, this section briefly describes each of the main components of the receiver, and points to their corresponding line numbers in the code. Most of the components can also be found in the block diagram in Fig. 1.

- **Calculate Intermediate Variables** (lines 18-25, 52-72) — This part calculates intermediate variables which are used in the receiver. This includes calculation of the effective carrier frequency and image frequency, memory allocation, variable allocation, and size determination.
- **Digital Downconversion via PLL** (lines 26-34, 74-82) — This part implements the downconversion which is accomplished in the current receiver with a PLL. The procedure consists of several sub-steps:
  - **Parameter Initialization and Bandpass Filter Design** (lines 26-34)
  - **PLL Pre-processing** (lines 74-76)
  - **PLL Adaptation** (lines 77-79)

  - **Mixing** (lines 80-82)

  Recall that the equation for PLL adaptation has the form

$$\theta[k+1] = \theta[k] - \mu x(kT_s)\sin\left(4\pi f_0 kT_s + \theta[k] + \psi\right)$$

  which appears in line 78. The carrier phase estimate is stored in the variable `theta` while the downconverted signal is stored in the variable `x_down`.

- **Pulse Matched Filter** (lines 48-51, 83-85) — This part performs filtering of the signal with the square-root raised-cosine filter. The filtered signal is stored in the variable `x_bb`.

- **Downsampling/Timing Sync via Output Power (OP) Method** (lines 35-38, 89-96) — This part performs the downsampling and timing synchronization using the method of output power maximization. The procedure makes several calls to the `interpsinc` function, and consists of several sub-steps:

  - **Parameter Initialization** (lines 35-38)
  - **Get current interpolated value** (line 90)
  - **Calculate approximate derivative** (lines 91-93)
  - **Algorithm Adaptation** (line 94)

  Recall the equation for the output-power-maximizing baud-timing adaptation algorithm has the form

$$\tau[k+1] = \tau[k] + \mu x[k]\left[x\left(\frac{kT}{M} + \tau[k] + \delta\right) - x\left(\frac{kT}{M} + \tau[k] - \delta\right)\right]$$

  which is seen in lines 91-94. The MATLAB variable `tau` stores the timing offset, `tnow` stores the current position, and `x_sampled` stores the downsampled signal after timing recovery.

- **Correlation** (lines 97-102) — Always running, this part calculates the correlation of the downsampled signal with the known header sequence.

- **Header Search** (lines 39-41, 104-110) — This part searches for the header, by comparing the correlation value with a threshold.

- **Equalization with Adaptation via LMS** (lines 111-124) — This part performs equalization of the signal, which includes adaptation of the equalizer coefficient during training periods (see paragraph below about different operating modes of receiver). Equalization consists of two sub-steps

  - **Adapt equalizer using LMS** (lines 111-121)
  - **Generate equalizer output** (lines 121-124)

Recall the equation for the LMS algorithm which has the form

$$f_i[k+1] = f_i[k] + \mu\,(s[k-\delta] - y[k])\,r[k-i]$$

and is seen in lines 114-115. The MATLAB variable `f` stores the equalizer coefficients, and `eqOut` stores the output of the equalizer.

- **Decision Device, Frame Sync, and Message Decoding** (lines 125-129) — This part quantizes the equalizer output using `quantalph`, resulting in a stream of 4-PAM symbol estimates stored in the variable `dec`. With knowledge of the start of the header sequence from the previous stage, frame synchronization is performed, after which the decisions pass into the decoder (i.e. `pam2letters`), the output of which is stored in the variable `decoded_msg`.

There are some other details of the receiver which are worth noting. The receiver consists of two main loops and their corresponding counters

1. `IFsampleIdx` — Each time this loop counter is incremented, the receiver has received a new IF sample at the receiver frontend.
2. `BBsampleIdx` — This loop counter is incremented every time a new baseband sample is output from the baud-timing device.

Also, the receiver operates in 3 distinct modes:

1. `HEADER_SEARCH_MODE` — In this mode, the receiver is running its correlator to search for the header sequence.
2. `TRAINING_MODE` — In this mode, the receiver thinks that it is receiving training data, and so it is training the equalizer using the LMS algorithm.
3. `DATA_MODE` — In this mode, the receiver has completed training, and believes that it is receiving data.

The receiver starts in `HEADER_SEARCH_MODE`. Once the header is found, it switches to `TRAINING_MODE`, and when the training is complete it switches to `DATA_MODE`. Once the data transmission is complete (based on the length of the data sequence specified in the system parameters), the receiver then returns to `HEADER_SEARCH_MODE` and repeats.

Listings

main.m

```
1 % Example script that demonstrates how to call transmitter and rece
  iver
2 % code.
3
```

```
% written by A. Klein 26-Oct-2005

% call script to get global simulation parameters (i.e. carrier freq, baud rate, training data, etc)
globalParams

% initialize random seed for repeatability (helps for debugging) --
randn('state',0);
rand('state',0);

% set channel, SNR, phase noise, and message to be sent -----------
c = [1 -0.4 0.2];              % channel (T-spaced)
SNR = 18;                      % signal-to-noise ratio (dB) of (under) sampled signal
phase_noise_variance = 1e-6;   % variance of underlying phase noise process

% (uncomment the following lines for the most benign conditions)
% c = [1 0 0];                % no ISI
% SNR = Inf;                  % no noise
% phase_noise_variance = 0;   % perfect oscillators

m=['This is the first frame which you probably shouldn''t able to decode perfectly unless you "cheat" and'
'give your receiver the initial points.  Now we''re into the second frame.  You might be able to decod'
'e this one, and now the third, error-free. . . But if you didn''t, then don''t worry yet.  The only fr'
'ames you are required to decode during the actual testing are those past the fifth frame.  So you''re'
'still okay.  We''re getting close to the end of the 5th frame, so your receiver better start working.'
'Congratulations!  If you can see this then you''re receiver has successfully decoded the fifth frame.'
'You might want to re-test your receiver by using different initial parameters, and different stepsiz'
'es to see what the effect is.  It''s probably help to plot the time history of the adaptive parameter'
'elements, too, so you can see if they''re taking too long to converge, if they seem unstable, etc.  A'
'nd now some more Nirvana lyrics: With the lights out it''s less dangerous Here we are now Entertain u'
's I feel stupid and contagious Here we are now Entertain us A mulatto An albino A mosquito My Libido'
'And I forget Just what it takes And yet I guess it makes me smile
```

```
    I found it hard Its hard to find. '
35  'Well, if your receiver has made it this far with no errors, and pe
    rforms error-free even when you ch'
36  'ange the initial parameter values, then it''s time to move on to t
    he "medium" test vector.  Good luck'];
37
38  % call transmitter ------------------------------------------------
39  [r, s]=Tx(m, c, SNR, phase_noise_variance);
40  m=m'; m=m(:)';
41
42  % call receiver -----------------------------------------------------
43  [decoded_msg y]=Rx(r);
44
45  % call code to calculate BER ----------------------------------
46  BERcalc
```

Tx.m

```
 1  function [r, s]=Tx(m, c, SNR, phase_noise_variance)
 2
 3  % function [r, s]=Tx(m, c, SNR, phase_noise_variance)
 4  %
 5  % Inputs:
 6  %                      m -- text message to be sent
 7  %                      c -- channel (T-spaced)
 8  %                    SNR -- signal to noise ratio
 9  %  phase_noise_variance -- variance of phase noise added to signal
10  %
11  % Outputs:
12  %   r -- received signal (at IF)
13  %   s -- transmitted symbols (for calculating SER)
14
15  % written by A. Klein 26-Oct-2005
16
17  global srrcLength marker training f_s T_t f_if rolloff dataLength
18
19  % determine suitable oversampling/downsampling factor
20  [M N]=rat(f_s*T_t);
21  M_scale=ceil((2*T_t*f_if+1+rolloff)/M);
22  M=M*M_scale;
23  N=N*M_scale;
24
25  % get dimensions
26  lines_of_text=size(m,1);
27  frame_length=(dataLength+(length([marker; training])));
```

```
28  char_str_length=frame_length*lines_of_text;
29
30  % insert training & header, and generate 4-PAM source vector
31  s=reshape([repmat([marker; training]',lines_of_text,1) reshape(lett
    ers2pam(reshape(m',lines_of_text*dataLength/4,1)),dataLength,lines_
    of_text)']',lines_of_text*(dataLength+(length([marker; training])))
    ,1);
32
33  % generate pulse-shaped signal
34  x=conv(srrc(srrcLength,rolloff,M,0)',upsample(s,M));
35
36  % mix signal to RF (analog upconversion)
37  p_noise=cumsum(randn(size(x))*sqrt(phase_noise_variance/N)); % gene
    rate phase noise process
38  x_rf=x.*cos(2*pi*f_if*[1:length(x)]'*T_t/M+p_noise);
39
40  % pass through BP channel
41  x2=conv(x_rf,upsample(c,M));
42
43  % add channel noise of appropriate SNR
44  x2_size=size(x2);
45  x2_nrm=sqrt(x2(srrcLength*M+1:x2_size(1)-srrcLength*M)'*x2(srrcLeng
    th*M+1:x2_size(1)-srrcLength*M)/x2_size(1));
46  x_r=x2+randn(size(x2))*10^(-SNR/20)*x2_nrm;
47
48  % perform analog conversion to IF, and do AGC
49  r=x_r(N:N:end);
50  r_nrm=r'*r/length(r);
51  r=r/sqrt(r_nrm);
52
53  % add some zeros to front and back
54  r=[zeros(floor(rand*10*M),1); r; zeros(10*M,1)];
```

Rx.m

```
 1  function [decoded_msg, eqOut]=Rx(r)
 2
 3  % function [decoded_msg, eqOut]=Rx(r)
 4  %
 5  % Inputs:
 6  %             r -- received signal (at IF)
 7  %
 8  % Outputs:
 9  %   decoded_msg -- received signal (at IF)
10  %   eqOut       -- output of equalizer
```

```
11
12 % written by A. Klein 26-Oct-2005
13
14 global srrcLength marker training f_s T_t f_if rolloff dataLength
15
16 globalParams
17
18 % calculate new effective carrier frequency, and the image which wi
   ll appear at 2f_c (and may get aliased)
19 f_c=f_if-fix(f_if/f_s)*f_s;
20 f_image=abs(mod(2*f_c+f_s/2,f_s)-f_s/2);
21
22 % calculate sizes ----------------------------
23 markerLength=length(marker);
24 trainingLength=length(training);
25
26 % PLL parameters & BPF filter design ----------------------------
27 bpf_ctr=f_image/f_s*2;    % set center frequency of BPF to 2f
28 BPFfilterOrder=500;       % should be an even number
29 ff=[0 bpf_ctr+[-0.006 -0.003 0.003 0.006]  1]; % BPF trans. band
30 fa=[0 0 1 1 0 0];              % values at transition regions
31 h=remez(BPFfilterOrder,ff,fa)';                    % design filter
32 phaseBPF=angle(exp(-1j*[0:length(h)-1]*pi*bpf_ctr)*h);  % calculate
    phase introduced by BPF
33 mu_PLL=0.001;             % stepsize for first PLL loop
34
35 % baud timing (OP) parameters ------------------------
36 mu_timing=0.1;                                  % algorithm stepsize
37 delta=0.1;                                     % time for derivative
38
39 % correlation (i.e. header search) parameters ---------
40 correlThresh=6500;                                  % threshold for
   determining whether we've received the header sequence
41
42 % equalizer parameters ---------------------------
43 eqLength=8;                            % equalizer length
44 mu_eq_lms=0.005;                       % trained LMS stepsize
45 eqDelay=3;                             % desired delay (for LMS)
46 f=zeros(eqLength,1); f(eqDelay+1)=1;  % equalizer initialization (m
   ust have correct length)
47
48 % design SRRC (matched) filter ---------------------------
49 srrcFlt=srrc(srrcLength,rolloff,f_s*T_t,0)';
50 srrcFltLength=length(srrcFlt);
51
```

```
% setup constants for each of the three operating modes------------
HEADER_SEARCH_MODE=1;
TRAINING_MODE=2;
DATA_MODE=3;
operationMode=HEADER_SEARCH_MODE;  % we start in HEADER_SEARCH_MODE

% allocate memory and initialize variables ---------------
theta=zeros(length(r),1);       % stores outputs of PLL
x_down=zeros(length(r),1);      % stores downconverted signal (pre-
matched filter)
x_bb=zeros(length(r),1);        % stores baseband signal (post-matc
hed filter)
x_sampled=zeros(ceil(length(r)/T_t/f_s),1);  % stores sampled signa
l (post timing recovery)
Corr=zeros(ceil(length(r)/T_t/f_s),1);       % stores correlation v
alues (for header search)
tau=zeros(ceil(length(r)/T_t/f_s),1);    % stores timing recovery (
only used for plotting)
eqOut=zeros(ceil(length(r)/T_t/f_s),1); % stores output of equalizr
e_lms=zeros(ceil(length(r)/T_t/f_s),1);      % stores LMS error
dec=zeros(ceil(length(r)/T_t/f_s),1);     % stores PAM-4 decisions
packetIndex=0;                               % packet counter
tnow=2*(srrcLength-2)*T_t*f_s;   % starting location for timing
BBsampleIdx=0;                  % intialize baseband sample counter
start=BPFfilterOrder+1;          % outer loop starting point

for IFsampleIdx=start:length(r);
    % pre-process signal for PLL
    r2(IFsampleIdx)=h'*r(IFsampleIdx:-1:IFsampleIdx-BPFfilterOrder)
.^2;

    % adapt PLL
    theta(IFsampleIdx+1)=theta(IFsampleIdx)-mu_PLL*r2(IFsampleIdx)*
sin(4*pi*f_c*IFsampleIdx/f_s+2*theta(IFsampleIdx)+phaseBPF);

    % perform downconversion
    x_down(IFsampleIdx)=r(IFsampleIdx)*cos(2*pi*f_c*IFsampleIdx/f_s
+theta(IFsampleIdx));

    % perform matched filtering
    x_bb(IFsampleIdx)=srrcFlt'*x_down(IFsampleIdx:-1:IFsampleIdx-sr
rcFltLength+1);

    while tnow<IFsampleIdx-2*srrcLength*T_t*f_s+2
 % do we have a new baseband sample?
```

```
87          BBsampleIdx=BBsampleIdx+1;
     % ok, we're at next baseband sample, so increment
88
89          % perform timing recovery (OP)
90          x_sampled(BBsampleIdx)=interpsinc(x_bb,tnow+tau(BBsampleIdx
    ),srrcLength);   % interpolated value at tnow+tau
91          x_deltap=interpsinc(x_bb,tnow+tau(BBsampleIdx)+delta,srrcLe
    ngth);           % get value to the right
92          x_deltam=interpsinc(x_bb,tnow+tau(BBsampleIdx)-delta,srrcLe
    ngth);           % get value to the left
93          dx=x_deltap-x_deltam;
                     % calculate numerical derivative
94          tau(BBsampleIdx+1)=tau(BBsampleIdx)+mu_timing*dx*x_sampled(
    BBsampleIdx);    % alg update: OP
95          tnow=tnow+T_t*f_s;
                     % update current position
96
97          % run correlator, matched to marker sequence
98          if (BBsampleIdx>eqDelay+markerLength-1)    % need to skip t
    he first few sample until we have enough to fill correlator
99              corInputSignal=x_sampled(BBsampleIdx-markerLength+1-eqD
    elay:BBsampleIdx-eqDelay);       % extract portion of signal used f
    or correlation
100             Corr(BBsampleIdx)=(marker'*corInputSignal)^2;
                                     % calculate correlation
101         end
102
103         switch operationMode
104             case HEADER_SEARCH_MODE     % if we haven't already fou
    nd marker, look for it...
105                 if Corr(BBsampleIdx)>correlThresh         % has cor
    relation exceeded threshold?
106                     operationMode=TRAINING_MODE;
     % yep, so switch to training mode
107                     trainingIndex=1;
     % reset to trainingIndex to first sample of training data
108                     packetIndex=packetIndex+1;
     % increment packet counter
109                 end 110
111             case TRAINING_MODE % if we're in equalizer training mod
    e, train the LMS equalizer
112
rr=x_sampled(BBsampleIdx:-1:BBsampleIdx-eqLength+1)
    ;                            % extract "regressor" vector of receive
    d signal
```

```
113                 eqOut(BBsampleIdx)=f'*rr;
                                % equalizer output
114
e_lms(BBsampleIdx)=training(trainingIndex)-eqOut(BB
    sampleIdx);                 % calculate LMS error term
115                 f=f+mu_eq_lms*e_lms(BBsampleIdx)*rr;
                                % update equalizer coefficients
116                 trainingIndex=trainingIndex+1;
                                % increment training index location
117                 if trainingIndex>trainingLength
                                % are we done training?
118                     operationMode=DATA_MODE;
                                    % yep, switch to data mode
119                     symbolIndex=1;
                                    % and re-init symbol counter to 1
120                 end
121
122             case DATA_MODE    % we're into data portion of the pack
    et -- equalizer, and save data
123
rr=x_sampled(BBsampleIdx:-1:BBsampleIdx-eqLength+1)
    ;                           % extract "regressor" vector of receive
    d signal
124                 eqOut(BBsampleIdx)=f'*rr;
                                % equalizer output
125
dec(symbolIndex)=quantalph(eqOut(BBsampleIdx),[-3 -
    1 1 3]);                    % make decisions
126
127                 if mod(symbolIndex,4)==0   % if we've completed a w
    hole letter (i.e. 4 PAM symbols), convert PAM symbols to letters
128
decoded_msg(packetIndex,symbolIndex/4)=pam2lett
    ers(dec(symbolIndex-3:symbolIndex)');  % re-assemble text message
129                 end
130
symbolIndex=symbolIndex+1;
                                % increment training index location
131                 if symbolIndex>dataLength  % are we done with data
    yet?
132                     operationMode=HEADER_SEARCH_MODE;  % yep, switc
    h back to header search mode
133                 end
134         end
135     end
```

```
136 end
137
138 % plot results ------------------------------------------------
139 figure(1);
140 plot(theta)
141 title('carrier phase estimate')
142 ylabel('theta')
143 xlabel('time')
144
145 figure(2)
146 plot(tau)
147 title('timing offset estimates')
148 ylabel('tau')
149 xlabel('time')
150
151 figure(3)
152 plot(Corr)
153 hold on
154 plot([1 length(Corr)],[correlThresh correlThresh],':')  % plot thre
    shold
155 title('correlator output (for finding start of training)')
156 xlabel('time')
157 ylabel('correlation value')
158
159 figure(4)
160 plot(eqOut,'b.')    % plot constellation diagram
161 title('constellation diagram (equalizer output)');
162 ylabel('estimated symbol values')
163 xlabel('time')
164
165 figure(5)
166 plot(e_lms)
167 title('error at equalizer output (during training)')
168 ylabel('e_lms')
169 xlabel('time')
```

globalParams.m

```
1 global srrcLength marker training f_s T_t f_if rolloff dataLength
2
3 srrcLength=4;              % truncated srrc length (divided by 2)
4 marker=letters2pam('0')';  % marker sequence
5 training=letters2pam('Oh well whatever Nevermind')';  % training se
  quence
6 f_s=850e3;                 % sampling frequency (Hz)
```

```
 7  T_t=6.4e-6;                % symbol period (seconds)
 8  f_if=2e6;                  % intermediate frequency (Hz)
 9  rolloff=0.3;               % srrc rolloff factor
10  dataLength=400;            % number of PAM symbols per data frame
```

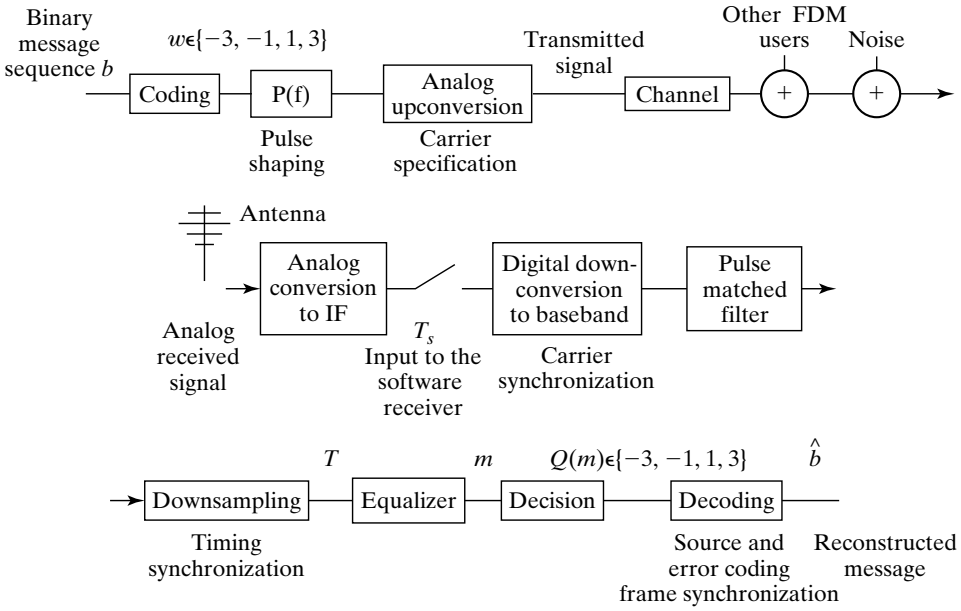
DAY 3

- Pulse Shaping and Receive Filtering
- Baud Timing for Clock Recovery

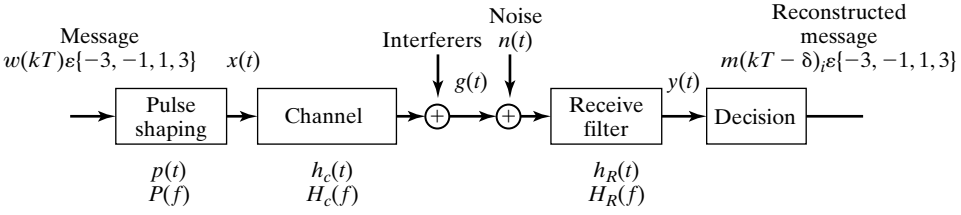
PULSE SHAPING AND RECEIVE FILTERING

- ⋆ Pulse and Pulse Amplitude Modulated Message Spectrum
- ⋆ Eye Diagram
- ⋆ Nyquist Pulses
- ⋆ Matched Filtering
- ⋆ Matched, Nyquist Transmit and Receive Filter Combination

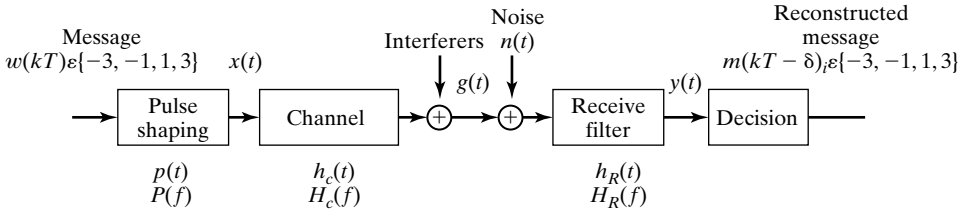
Pulse Shaping and Receive Filtering



We will focus on the situation where up and downconversion have been flawlessly performed and the effect of transmission from baseband PAM message waveform to received signal is presumed described by a linear transfer function and the addition of interferers, in particular spectrally flat broadband noise.



Pulse and pulse amplitude modulated (PAM) message spectrum



The spectral footprint of a baseband PAM signal is no wider than that of the pulse shape.

- Compose the analog pulse train entering the pulse shaping filter as

$$w_a(t) = \sum_k w(kT)\delta(t - kT)$$

  which is $w(kT)$ for $t = kT$ and 0 for $t \neq kT$

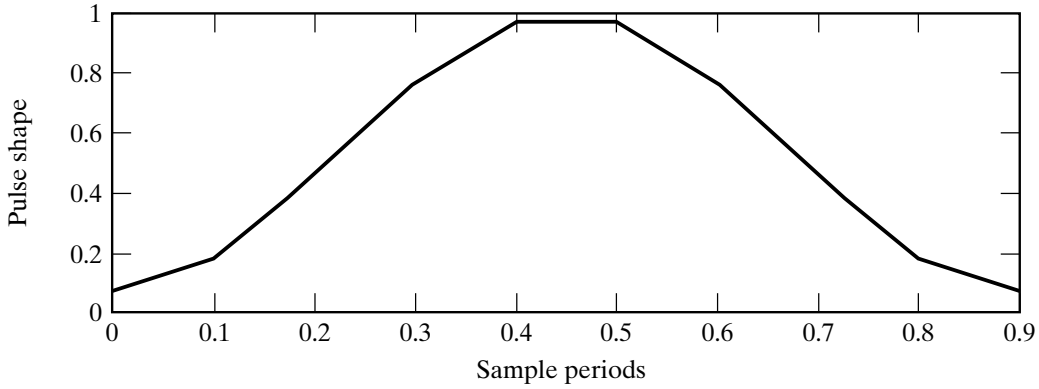
- Pulse shaping filter output

$$x(t) = w_a(t) * p(t) \quad \Rightarrow \quad X(f) = W_a(f)P(f)$$

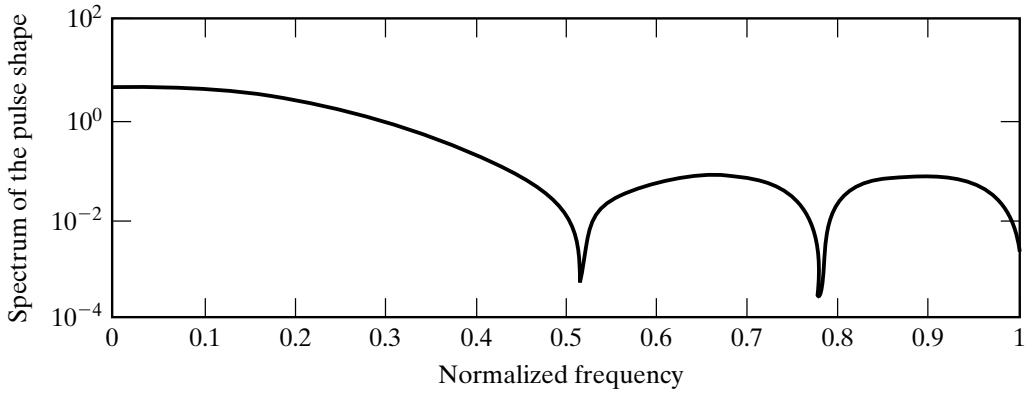
- $X(f)$ cannot be nonzero at frequencies where $P(f)$ is zero.

Pulse ... message spectrum (cont'd)

One-symbol wide Hamming blip pulse shape (with 10 samples per symbol) and frequency response (using `freqz` in `pulsespec`)



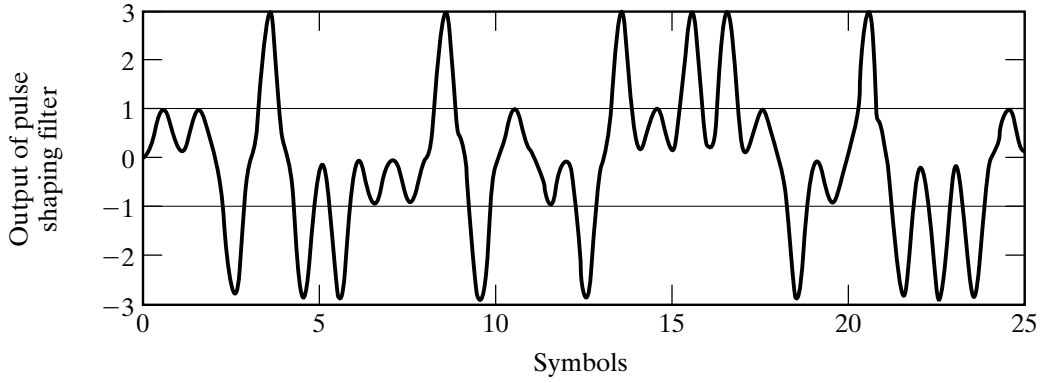
(a)

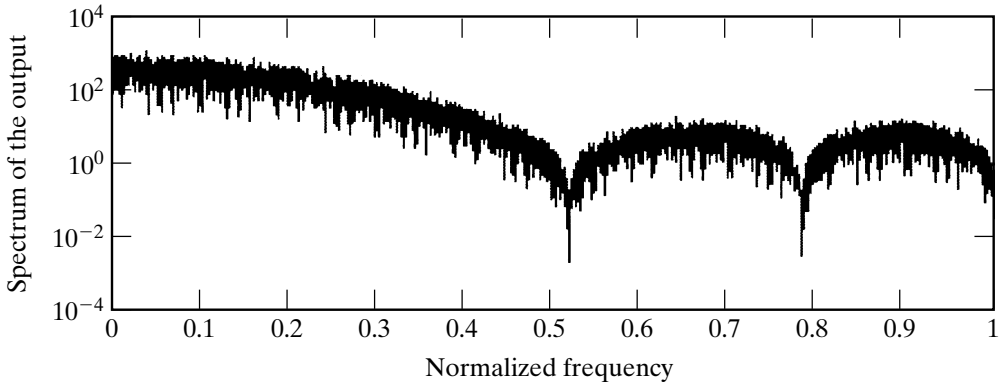


(b)

Pulse ... message spectrum (cont'd)

Spectrally flat 4-PAM symbol sequence triggering baud-spaced 10-times oversampled Hamming blip pulse shape as (baseband) output of pulse shaping filter



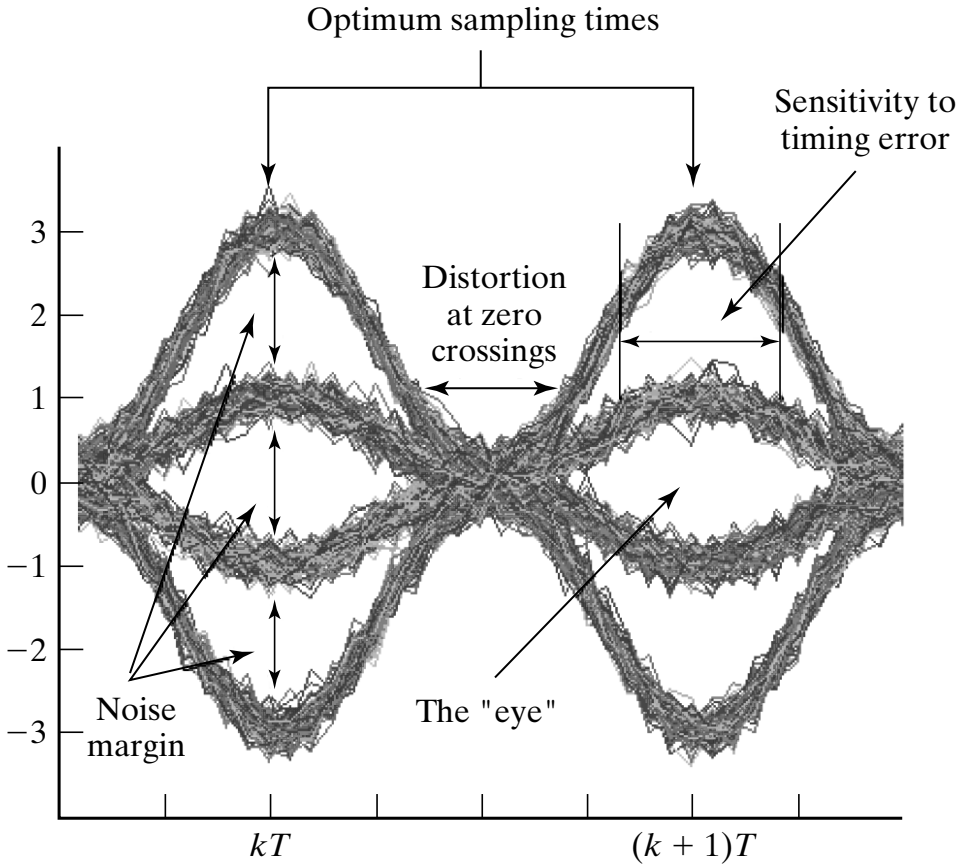


Message signal spectrum has scalloped contours of Hamming blip pulse frequency response.

Eye Diagram

Eye diagram is a popular robustness evaluation tool.

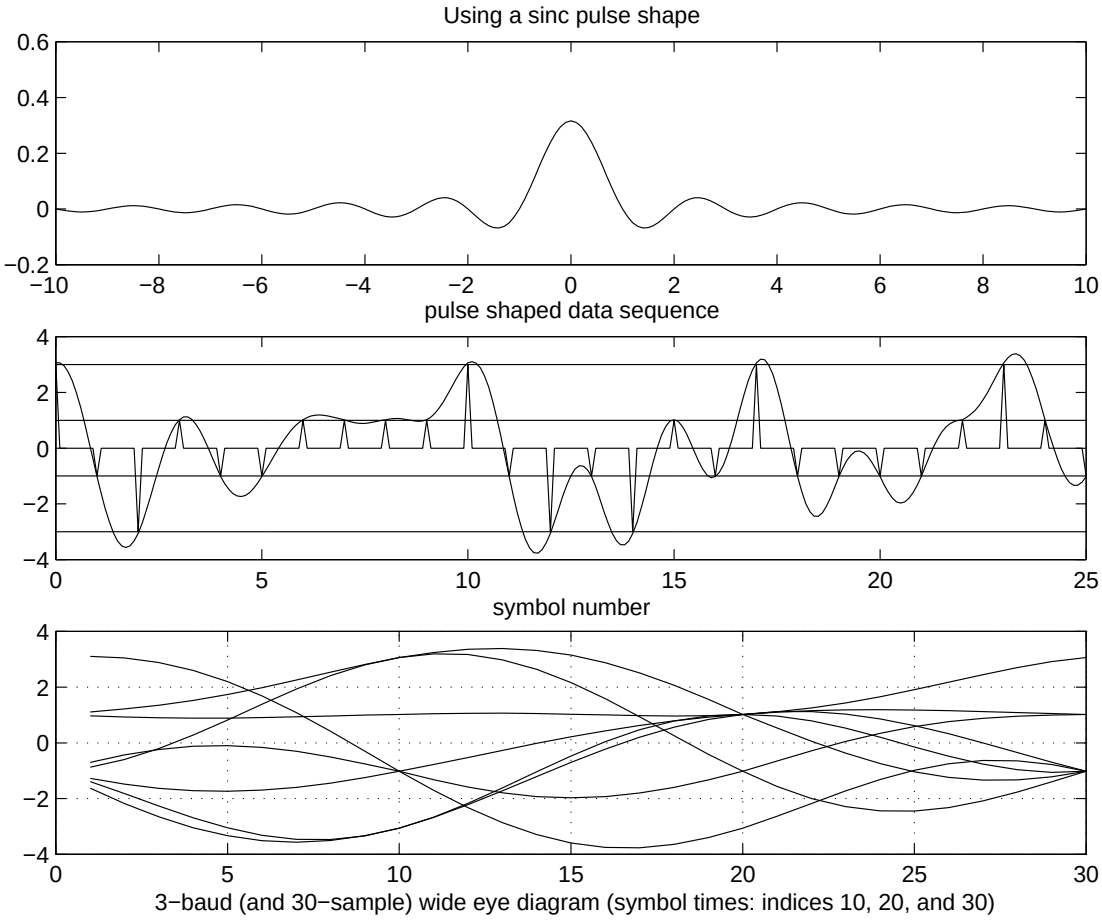
For 4-PAM, single-baud-wide Hamming blip with additive broadband channel noise, retriggering oscilloscope after every 2 baud intervals produces



Observe illustrative vertical (amplitude) and horizontal (timing) margins for correct decision at sample times.

Eye Diagram (cont'd)

Consider 20-symbol wide, 10 times oversampled, truncated, sinc pulse $(\sin(\pi t/T)/(\pi t/T))$ with zero-crossings at kT for $k = 1, 2, ..., 10$ for 4-PAM symbol sequence (from `spsex`)


Using a sinc pulse shape
pulse shaped data sequence
symbol number
3−baud (and 30−sample) wide eye diagram (symbol times: indices 10, 20, and 30)

A multi-baud-wide pulse shape, but no ISI!

Nyquist Pulses

The impulse response of a Nyquist pulse creating no ISI at other sample times is zero at those instants and nonzero only at the one particular sample time.

- The impulse response $p(t)$ is a Nyquist pulse for a T-spaced symbol sequence if there exists a τ such that

$$p(t)|_{t=kT+\tau} = \begin{cases} c, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

- Rectangular pulse:

$$p_R = \begin{cases} 1, & 0 \leq t < T \\ 0, & \text{otherwise} \end{cases}$$

  - ⊙ Rectangle is Nyquist pulse for almost any sampler timing.

Nyquist Pulses (cont'd)

- Sinc pulse:

$$p_S(t) = \frac{\sin \pi f_0 t}{\pi f_0 t}$$

  where $f_0 = 1/T$.

  - ⊙ Sinc is Nyquist pulse because $p_S(0) = 1$ and $p_S(kT) = \frac{\sin(\pi k)}{\pi k} = 0$.
  - ⊙ Sinc envelope decays at $1/t$.

- Raised-cosine pulse:

$$p_{RC}(t) = 2f_0 \left(\frac{\sin(2\pi f_0 t)}{2\pi f_0 t} \right) \left[\frac{\cos(2\pi f_\Delta t)}{1 - (4 f_\Delta t)^2} \right]$$

  with roll-off factor $\beta = f_\Delta / f_0$.

  - ⊙ Raised-cosine is Nyquist pulse for $T = 1/2f_0$ because p_{RC} has a sinc factor $\sin(\pi k)/\pi k$ which is zero for all nonzero integers k.
  - ⊙ Raised-cosine envelope decays at $1/|t^3|$.
  - ⊙ As $\beta \to 0$, raised-cosine $\to$ sinc.

Nyquist Pulses (cont'd)

- Raised-cosine pulse (cont'd):
  - ⊙ Fourier transform

$$P_{RC}(f) = \begin{cases} 1, & |f| < f_1 \\ \frac{1+\cos(\alpha)}{2}, & f_1 < |f| < B \\ 0, & |f| > B \end{cases}$$

where

B is the absolute bandwidth,

f_0 is the 6db bandwidth,

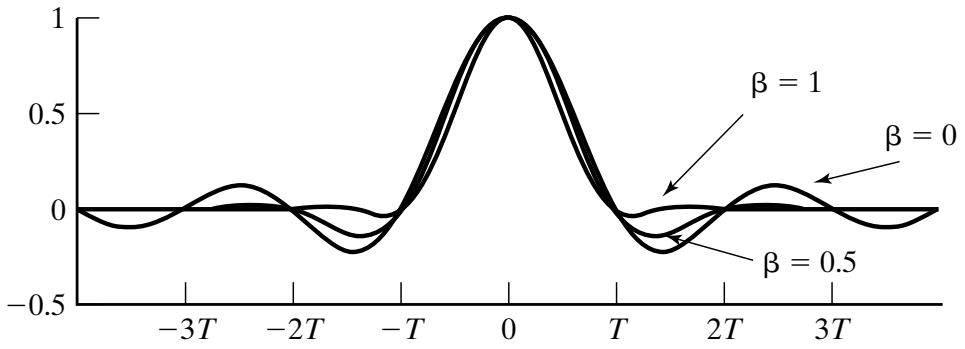
$f_\Delta = B - f_0$,

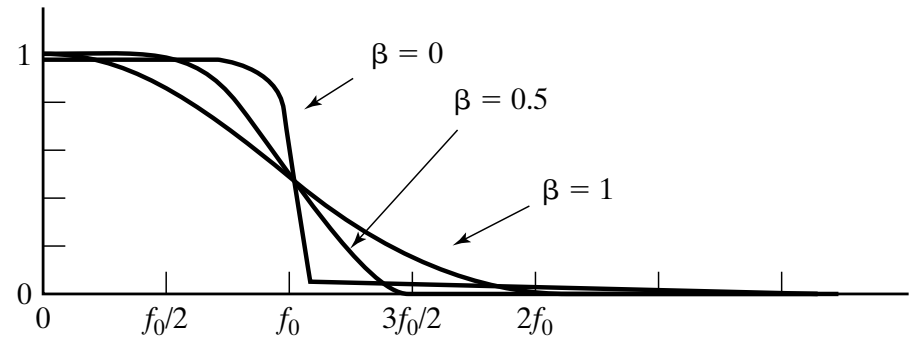
$f_1 = f_0 - f_\Delta$, and

$\alpha = \frac{\pi(|f|-f_1)}{2f_\Delta}$

Nyquist Pulses (cont'd)

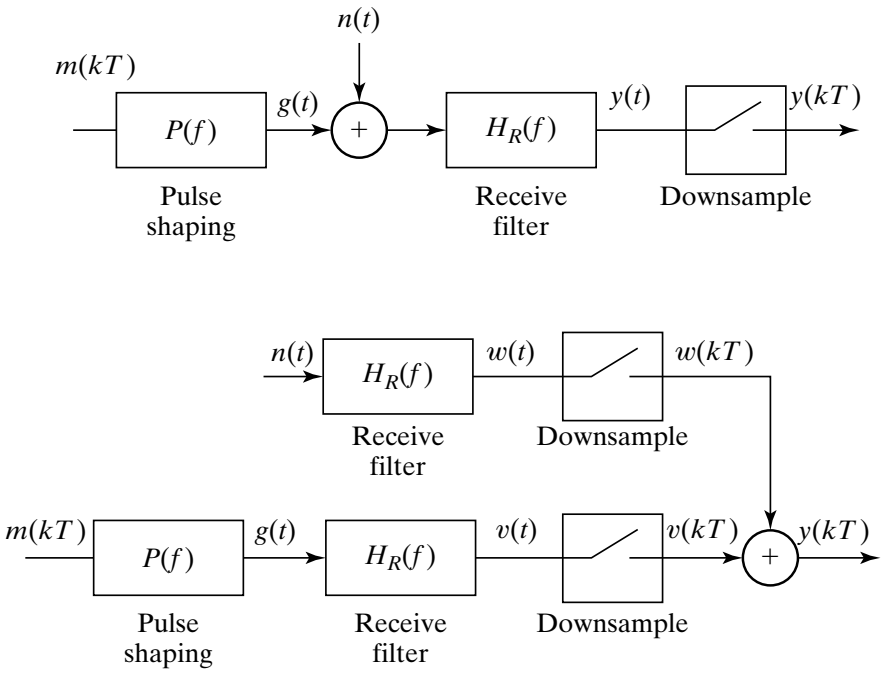
- Raised-cosine pulse (cont'd):
  - ⊙ Time and Frequency Plots:





Matched Filter

Suppose the channel simply adds broadband noise $n(t)$. The symbol to reconstructed downsample system is described by



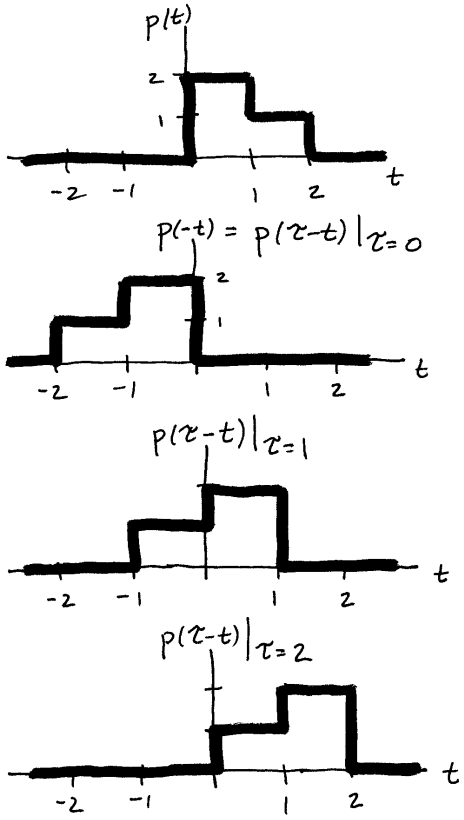
so $y(t) = v(t) + w(t) = h_R(t) * g(t) + h_R(t) * n(t)$.

- Our objective is to choose $h_R(t)$ to maximize the power of the signal $v(t)$ at a specific time $t = \tau$, i.e. $v^2(\tau)$, relative to the total power of $w(t)$ where the power spectral density of $n(t)$ is a constant η over all frequencies.

Matched Filter (cont'd)

With spectrally flat channel noise the SNR-maximizing receive filter impulse response is the time-reversal of that of the pulse shape.

- Example:



  Minimum τ for causality of matched filter is pulse width for pulse initiated at $t = 0$.

- NOTE: Minimum-delay matched filter is same as pulse if pulse is causal and even symmetric.

Matched Nyquist Transmit and Receive Filter Combinations

A preferred receive filter impulse response (in the absence of channel ISI but with broadband channel noise) (i) will match the reversed impulse response of the transmitter pulse shape and (ii) when convolved with the transmitter pulse shape will form a Nyquist pulse.

- Want convolution of candidate pulse shape $g(t)$ and its matched filter $g(t-\tau)$ to equal even symmetric Nyquist pulse $p(t)$.
- Since convolution of two even symmetric pulse shapes is even symmetric, presume $g(t)$ is even symmetric, so with particular τ, $g(t) = g(\tau - t)$.
- Objective becomes

$$p(t) = g(t) * g(t) \quad \Rightarrow \quad P(f) = G^2(f)$$

Matched ... Combinations (cont'd)

- So, choose

$$G(f) = \sqrt{P(f)} \quad \Rightarrow \quad g(t) = \mathcal{F}^{-1}\{\sqrt{P(f)}\}$$

- For example, consider the square-root raised cosine (SRRC)

$$v(t) = \begin{cases} \frac{1}{\sqrt{T}} \frac{\sin(\pi(1-\alpha)t/T)+(4\alpha t/T)\cos(\pi(1+\alpha)t/T)}{(\pi t/T)(1-(4\alpha t/T)^2)} \\ \qquad \text{for} \;\; t \neq 0, \; t \neq \pm\frac{T}{4\alpha} \\ \frac{1}{\sqrt{T}}(1-\alpha+(4\alpha/\pi)) \quad \text{for } t = 0 \\ \frac{\alpha}{\sqrt{2T}}\left[\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4\alpha}\right)+\left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4\alpha}\right)\right] \\ \qquad \text{for } t = \pm\frac{T}{4\alpha} \end{cases}$$

  which has a magnitude spectrum the square of which equals the magnitude spectrum of a raised cosine.

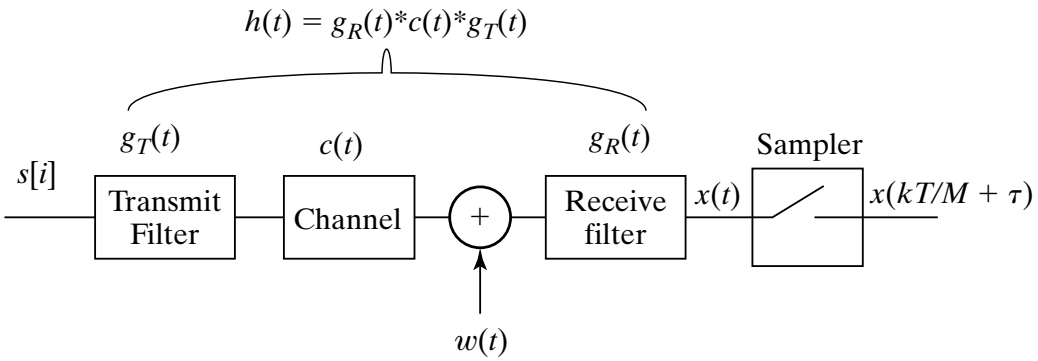
- The square root raised cosine is the most commonly used pulse in bandwidth constrained communication systems.

BAUD TIMING FOR CLOCK RECOVERY

- ⋆ A Baud-Timing Example
- ⋆ Output Power Maximization

Baud-Timing

- Consider the situation where the up and down conversion is done perfectly, so we need only consider a baseband model of the communication system.



- We are to select τ in

$$x[k] = x(\frac{kT}{M} + \tau)$$

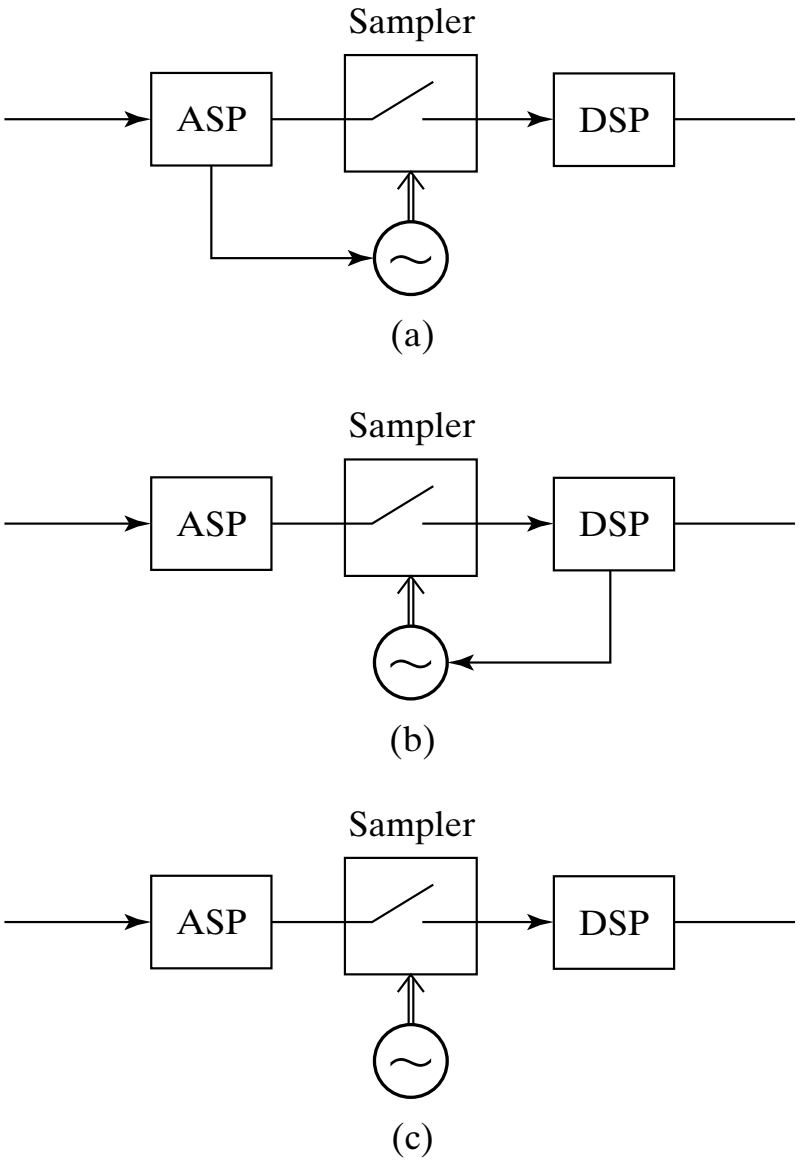
$$= \left(\sum_{i=-\infty}^{\infty} s[i]h(t - iT) + w(t) * g_R(t) \right) |_{t=\frac{kT}{M}+\tau}$$

with

$$h(t) = g_T(t) * c(t) * g_R(t)$$

Baud-Timing (cont'd)

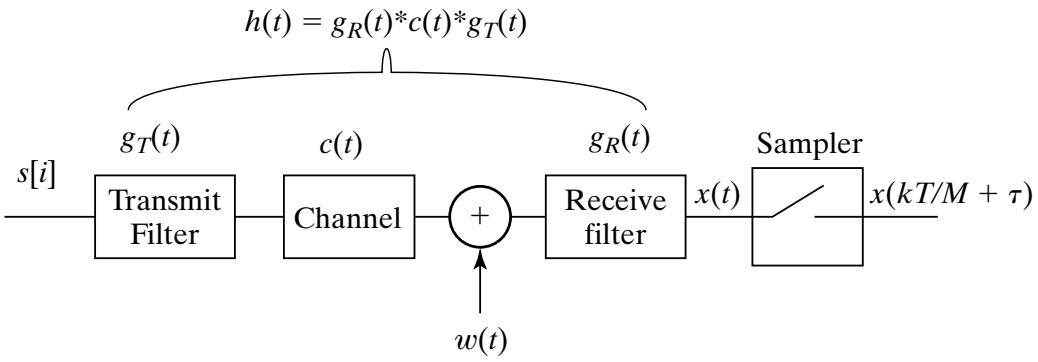
Three possible implementation configurations



We favor the last with its free-running sampler and fine tuning of the baud-timing done in the receiver DSP.

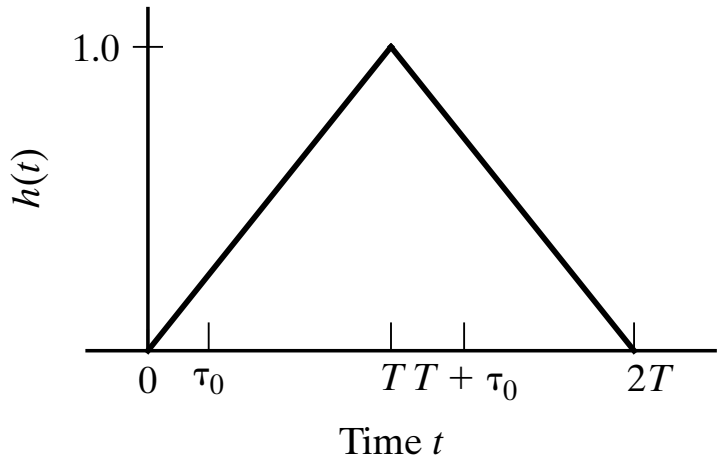
A Baud-Timing Example

We will analyze the special case for



when

- the noise w is absent and
- the analog pulse-shaping filter, the channel transfer function, and the receive filter combine into an impulse response that is a triangle spanning two symbol intervals.



A Baud-Timing Example (cont'd)

- With perfect baud-timing ($\tau = 0$) baud-space-sampled ($M = 1$) combined analog pulse/channel/receive filter impulse response shape is a Nyquist pulse

$$h(kT) = \begin{cases} 1, & k = 1 \\ 0, & k \neq 1 \end{cases}$$

- In general, without perfect baud-timing the sampler output is a weighted combination of several source symbol values

$$x[k] = \sum_i s[i] h(t - iT) \Big|_{t=kT+\tau}$$

- Consider three cases:
  - ⊙ $\tau = 0$
  - ⊙ $\tau > 0$
  - ⊙ $\tau < 0$

A Baud-Timing Example (cont'd)

- $\tau = 0$
  - ⊙ Only one nonzero point in sampled impulse response
  - ⊙ Sampled impulse response

$$\begin{aligned} h(t-iT)|_{t=kT+\tau} &= h(kT+\tau-iT) \\ &= h((k-i)T+\tau) \\ &= h((k-i)T) \\ &= \begin{cases} 1, & k-i=1 \\ & \Rightarrow\ i=k-1 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

  - ⊙ $x[k] = s[k-1]$, system is pure delay, and sampler is synchronized with transmitter pulse.

A Baud-Timing Example (cont'd)

- $\tau > 0$
  - ⊙ Two nonzero points in sampled impulse response $h(\tau_0)$ and $h(T + \tau_0)$
  - ⊙ Sampled impulse response

$$h(t - iT)|_{t=kT+\tau_0} = h((k - i)T + \tau_0)$$

$$= \begin{cases} 1 - \frac{\tau_0}{T}, & k - i = 1 \\ \frac{\tau_0}{T}, & k - i = 0 \\ 0, & \text{otherwise} \end{cases}$$

- $\tau < 0$
  - ⊙ Two nonzero points in sampled impulse response $h(2T + \tau_0)$ and $h(T + \tau_0)$.
  - ⊙ Sampled impulse response

$$h(t - iT)|_{t=kT+\tau_0} = \begin{cases} 1 - \frac{|\tau_0|}{T}, & k - i = 1 \\ \frac{|\tau_0|}{T}, & k - i = 2 \\ 0, & \text{otherwise} \end{cases}$$

A Baud-Timing Example (cont'd)

Any sampled output $x[k]$ is based only on, at most, two symbol-spaced samples for any choice of τ.

- For example, with $\tau > 0$ for $k = 6$

$$x[6] = \sum_i s[i]h((6-i)T + \tau_0)$$

$$= s[6]h(\tau_0) + s[5]h(T + \tau_0)$$

$$= s[6]\frac{\tau_0}{T} + s[5](1 - \frac{\tau_0}{T})$$

- For example, with $\tau < 0$ for $k = 6$

$$x[6] = \sum_i s[i]h((6-i)T + \tau_0)$$

$$= s[5]h(T + \tau_0) + s[4]h(2T + \tau_0)$$

$$= s[4]\frac{|\tau_0|}{T} + s[5](1 - \frac{|\tau_0|}{T})$$

- For a binary input there are 4 possible symbol pairs $(+1, +1)$, $(+1, -1)$, $(-1, +1)$, and $(-1, -1)$ that are assumed equally likely.

A Baud-Timing Example (cont'd)

- For example, with $\tau > 0$ for $k = 6$
  - $\odot$ $(s[5], s[6]) = (+1, +1) \Rightarrow$
    $x[6] = \frac{\tau_0}{T} + 1 - \frac{\tau_0}{T} = 1$
  - $\odot$ $(s[5], s[6]) = (+1, -1) \Rightarrow$
    $x[6] = \frac{-\tau_0}{T} + 1 - \frac{\tau_0}{T} = 1 - \frac{2\tau_0}{T}$
  - $\odot$ $(s[5], s[6]) = (-1, +1) \Rightarrow$
    $x[6] = \frac{\tau_0}{T} - 1 + \frac{\tau_0}{T} = -1 + \frac{2\tau_0}{T}$
  - $\odot$ $(s[5], s[6]) = (-1, -1) \Rightarrow$
    $x[6] = \frac{-\tau_0}{T} - 1 + \frac{\tau_0}{T} = -1$
- Two of the possibilities for $x[6]$ give correct values for $s[5]$, while two are incorrect.
- As long as $2\tau_0 < T$ then the sign$[x(6)]$ matches $s[5]$ for all four possibilities.
- If τ_0 exceeds $T/2$, the sign of $x(6)$ would be associated with an earlier s than $s[5]$.

A Baud-Timing Example (cont'd)

- Similarly, with $\tau < 0$ for $k = 6$, the four equally likely source symbol pairs creating $x[6]$ are
  - $\odot$ $(s[4], s[5]) = (+1, +1) \Rightarrow$
    $x[6] = \frac{|\tau_0|}{T} + 1 - \frac{|\tau_0|}{T} = 1$
  - $\odot$ $(s[4], s[5]) = (+1, -1) \Rightarrow$
    $x[6] = \frac{-|\tau_0|}{T} + 1 - \frac{|\tau_0|}{T} = 1 - \frac{2|\tau_0|}{T}$
  - $\odot$ $(s[4], s[5]) = (-1, +1) \Rightarrow$
    $x[6] = \frac{|\tau_0|}{T} - 1 + \frac{|\tau_0|}{T} = -1 + \frac{2|\tau_0|}{T}$
  - $\odot$ $(s[4], s[5]) = (-1, -1) \Rightarrow$
    $x[6] = \frac{-|\tau_0|}{T} - 1 + \frac{|\tau_0|}{T} = -1$
- With the addition of the absolute value on τ_0 (which does not effect a positive τ_0) the formulas for the four choices are the same as for positive τ_0.

A Baud-Timing Example (cont'd)

- For $-T/2 < \tau_0 < T/2$, $\mathrm{Q}(x[k]) = s[k-1]$.
- So, source recovery error equals decision error

$$e[k] = s[k-1] - x[k] = \mathrm{Q}(x[k]) - x[k]$$

  when eye is open. (But, if eye is closed, cluster variance does not equal average squared recovery error.)

- We are now in a position to consider some candidate cost functions for this baud-timing example.

A Baud-Timing Example (cont'd)

- Cluster variance

$$\text{avg}\{(\text{Q}(x[k]) - x[k])^2\}$$

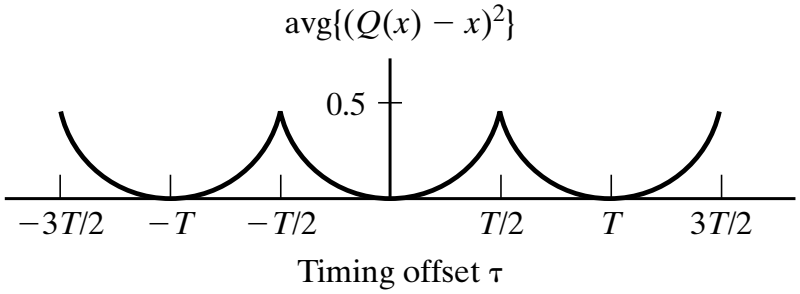
- $\text{avg}\{(\text{Q}(x[6]) - x[6])^2\}$

$$= \left\{(1-1)^2 + (1-(1-\frac{2|\tau_0|}{T}))^2 \right.$$

$$\left. +(-1-(-1+\frac{2|\tau_0|}{T}))^2 + (-1-(-1))^2 \right\}$$

$$= \left(\frac{1}{4}\right)\left(\frac{4\tau_0^2}{T^2} + \frac{4\tau_0^2}{T^2}\right) = \frac{2\tau_0^2}{T^2}$$

- The same result occurs for other k.
- Desired offset of $\tau = 0$ $(\pm nT)$ occurs with minimization of average squared decision error in the sampler output



A Baud-Timing Example (cont'd)

- Average squared sampler output (or output power)

  $\text{avg}\{x^2[k]\}$

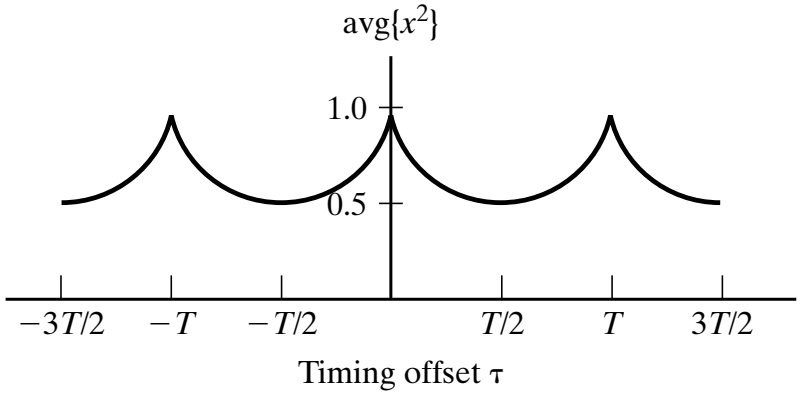
$$= (1/4)[(1)^2 + (1 - (2|\tau|/T))^2$$

$$+(-1 + (2|\tau|/T))^2 + (-1)^2]$$

$$= (1/4)[2 + 2(1 - (2|\tau|/T))^2]$$

$$= 1 - (2|\tau|/T) + (2|\tau|^2/T^2)$$

- Desired offset of $\tau = 0$ $(\pm nT)$ occurs with maximization of average squared sampler output



Output Power Maximization

- Moving average of square of sampler output

$$J_{OP}(\tau) = \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} \{x^2[k]\} = \text{avg}\{x^2[k]\}$$

- To maximize J_{OP} using a gradient ascent

$$\tau[k+1] = \tau[k] + \bar{\mu} \frac{\partial}{\partial \tau} [\text{avg}\{x^2[k]\}]|_{\tau=\tau[k]}$$

with small $\bar{\mu}$, we interchange the average and the differentiation and drop the "outer" average yielding

$$\tau[k+1] = \tau[k] + \bar{\mu} \frac{\partial (x^2[k])}{\partial \tau}|_{\tau=\tau[k]}$$

$$= \tau[k] + 2\bar{\mu} \left(x[k] \frac{\partial x[k]}{\partial \tau} \right) |_{\tau=\tau[k]}$$

where for small δ

$$\frac{dx[k]}{d\tau} = \frac{dx(\frac{kT}{M}+\tau)}{d\tau}$$

$$\approx \frac{x(\frac{kT}{M} + \tau + \delta) - x(\frac{kT}{M} + \tau - \delta)}{2\delta}$$

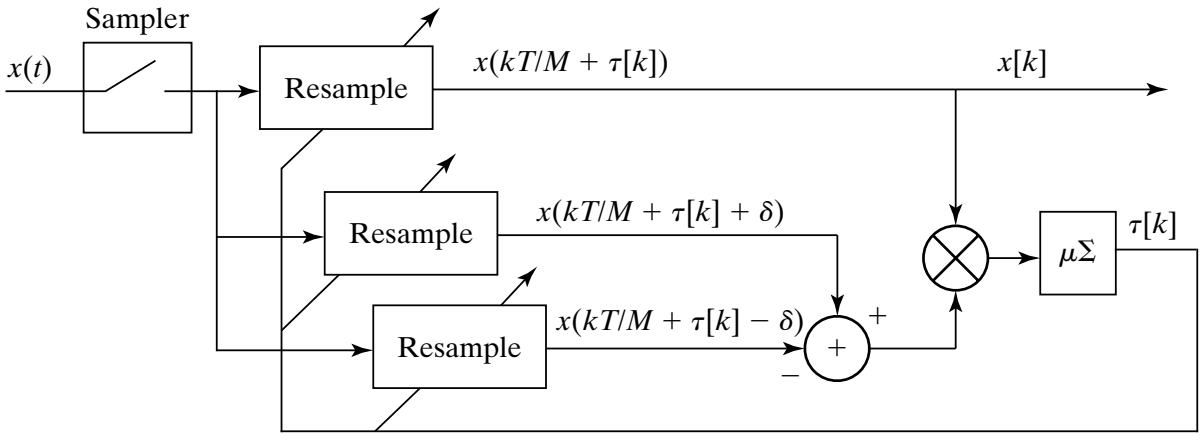
Output Power Maximization (cont'd)

- Output-power-maximizing baud-timing adaptation algorithm (with $x[k] = x((kT/M) + \tau[k]))$

$$\tau[k+1] = \tau[k] + \mu x[k]$$

$$\cdot \left(x(\frac{kT}{M} + \tau[k] + \delta) - x(\frac{kT}{M} + \tau[k] - \delta) \right)$$

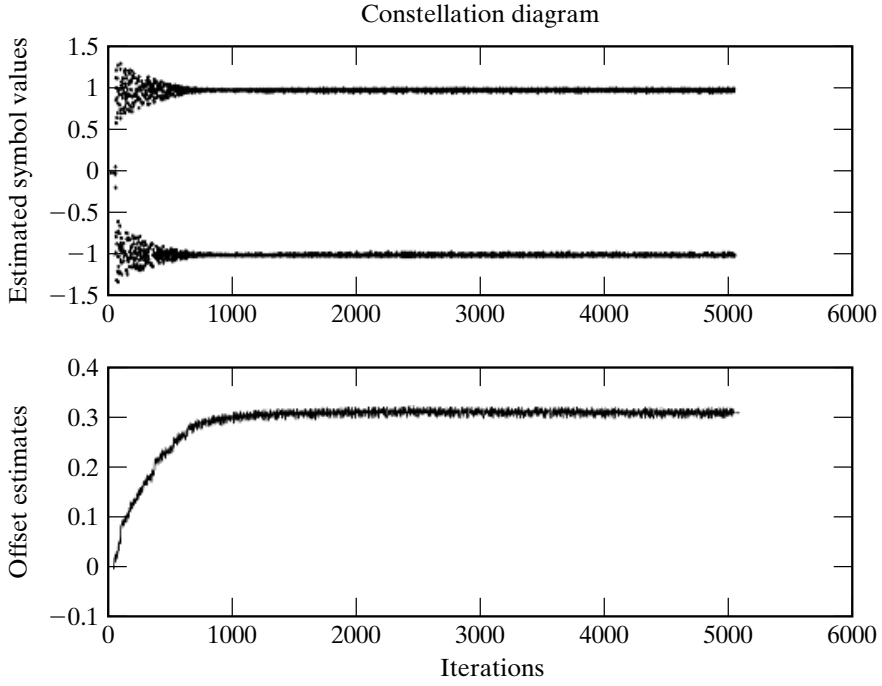
- Output-power-maximizing baud-timing adjusted oversampler schematic



Output Power Maximization (cont'd)

Example (from `clockrecOP`*)*:

- Source: 2-PAM
- Baud-timing adaptor stepsize: $\mu = 0.05$
- Derivative approximation increment: $\delta = 0.1$
- Pulse shape: SRRC with $\beta = 0.5$
- Free-running receiver sampler offset: -0.3 ($\Rightarrow$ desired baud-timing adjustment of $\tau = 0.3$)

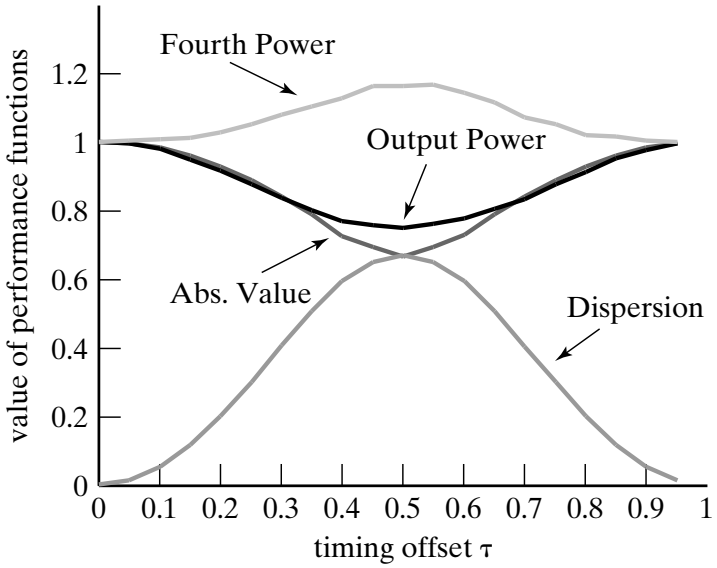


Output Power Maximization (cont'd)

Example (from `clockrecOPcost`*)*:

Cost functions for desired τ of zero with SRRC pulse shape with roll-off factor $\beta = 0.5$

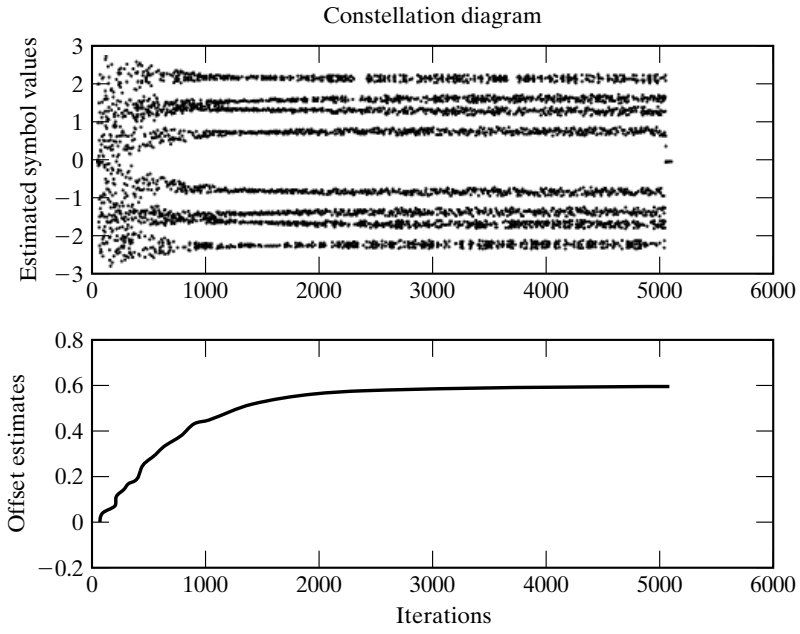
- absolute value: $J_{AV} = \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} \{|x[k]|\}$
- fourth power: $J_{FP} = \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} \{x^4[k]\}$
- output power (aka output energy): $J_{OP}(\tau) = \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} \{x^2[k]\}$
- dispersion (aka constant modulus): $J_D(\tau) = \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} \{(x^2[k] - 1)^2\}$



Output Power Maximization (cont'd)

What happens with ISI? (using `clockrecOP`*)*:

- Channel: [1, 0.7, 0, 0, 0.5]
- All else same. (2-PAM source; $\mu = 0.05$; $\delta = 0.1$; SRRC pulse with $\beta = 0.5$; Free-running receiver sampler offset: -0.3; and $M = 2$)



- Initially closed eye is opened within 500 iterations.
- Asymptotic offset not same as without ISI.

Laboratory Exercises – Day 3

Introduction to Digital Communication Receiver Design

Task 1: Algorithm Derivation for Dispersion-Minimization-Based Baud-Timing

In the lecture notes, you were shown how to implement the Output-Power-Maximization (OP) technique for clock recovery. Recall that the cost function to maximize in the OP technique is given by

$$J_{OP}(\tau) \;=\; \text{avg}\left\{x^2[k]\right\}$$

which can be implemented via steepest ascent, resulting in the adaptation algorithm

$$\tau[k+1] \;=\; \tau[k] + \mu x[k]\left(x\left(\frac{kT}{M} + \tau[k] + \delta\right) - x\left(\frac{kT}{M} + \tau[k] - \delta\right)\right)$$

Your task: Derive the steepest descent baud-timing algorithm for the dispersion minimization (DM) cost given by

$$J_{DM}(\tau) \;=\; \text{avg}\left\{\left(x^2[k]-1\right)^2\right\}$$

You may find the discussion on the first two pages of the "Output Power Maximization" section of the DAY 2 lecture notes to be useful.

Task 2: Implementation of Dispersion-Minimization-Based Baud-Timing

The receiver you have been given currently uses the OP technique for clock recovery (in `/system_code/Rx.m`). Your task is to replace the OP algorithm with the dispersion-minimization algorithm you developed above. You will then compare the performance of the two schemes.

After you have successfully added the DM algorithm to the system, you should tune its parameters (stepsize, etc) for best performance. Lastly, you should run `/system_code/main.m` to see how the receiver performs in comparison to the old OP-based receiver. You should examine a plot of the timing offset estimate, τ, when answering the following questions:

1. Consider a case with no noise, and no phase noise, but with a channel having taps $c = [-0.7, 1, 0.4, 0.2]$. To what value of `tau` does the DM algorithm converge to? To what value of `tau` does the OP algorithm converge to? Do both algorithms converge to the same value? Include a plot for each.

2. Decrease the SNR in 3 dB steps, and determine which algorithm first begins to make bit errors.

3. What effect do you observe as you decrease the stepsize? What happens as you increase the stepsize?

4. Change the SRRC rolloff factor (in `/system_code/globalParams.m`). What is the effect on algorithm performance when you increase or decrease the rolloff factor?

DAY 4

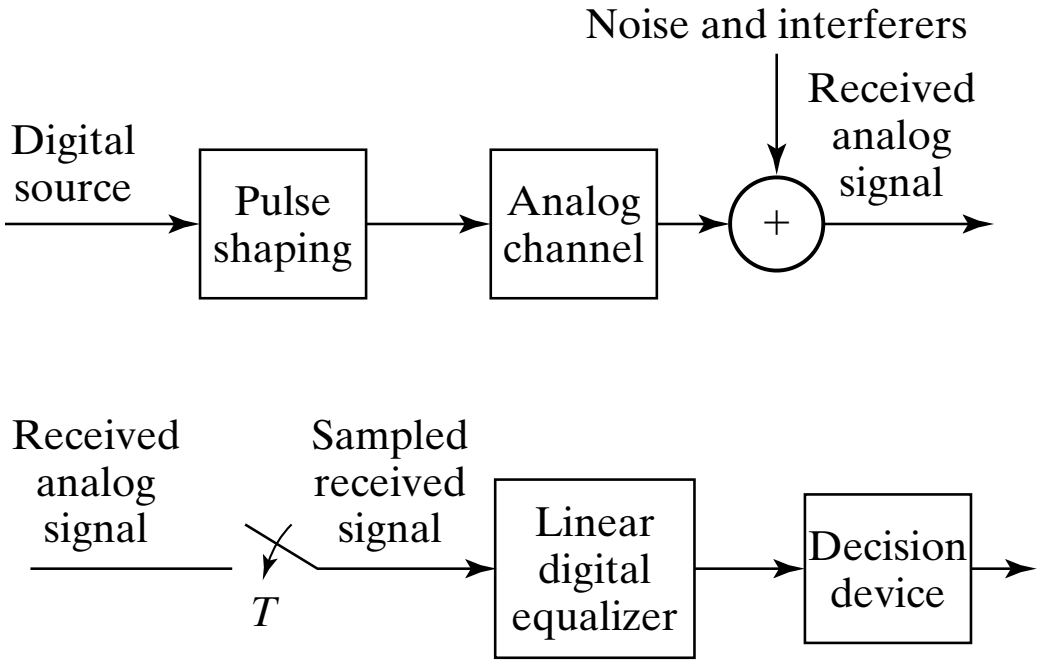
- Linear Equalization
- Putting It All Together: Receiver Design

LINEAR EQUALIZATION

- ⋆ Multipath and Other Interference
- ⋆ Trained Linear Equalization
- ⋆ Trained Adaptive Least-Mean-Square Equalization
- ⋆ Blind Adaptive Decision-Directed Equalization
- ⋆ Blind Adaptive Dispersion Minimizing Equalization

Multipath and Other Interference

- Assume up and down conversion and carrier and clock recovery (including matched filtering and downsampling) all executed transparently.
- Impairment of interest is multipath interference (linear filtering by analog channel and receiver front-end preceding equalizer) and other additive interference (broadband noise and narrowband interferers).



Multipath ... Interference (cont'd)

- FIR channel model:

$$y(kT) = a_1 u(kT) + a_2 u((k-1)T)$$

$$+ \ldots + a_n u((k-n)T) + \eta(kT)$$

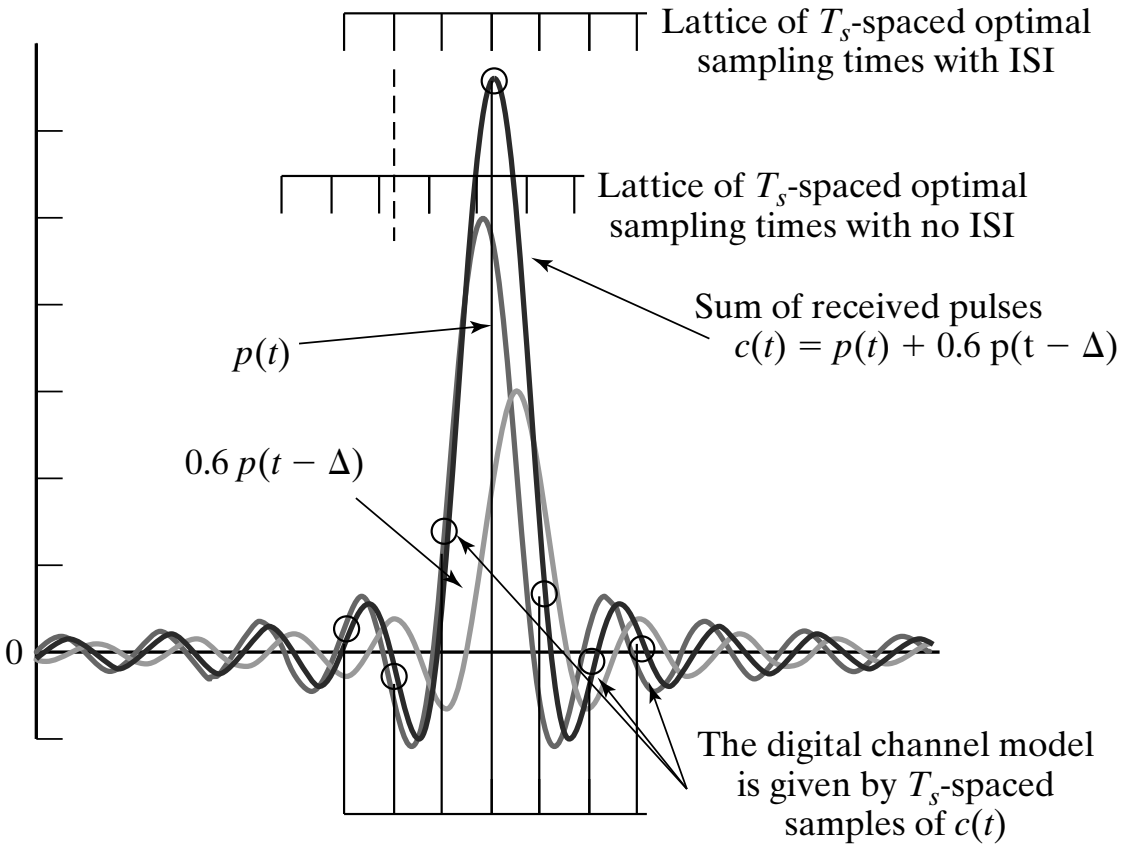
  where $\eta(kT)$ is sample of other interference.

- Order n of discrete-time FIR channel model dependent on physical delay spread of channel.

- For 4 μsec delay spread by "physical" channel:
  - $\odot$ $T = 0.04$ μsec $\rightarrow$ 25 Msymbols/sec $\rightarrow$ $n = 100$
  - $\odot$ $T = 0.4$ μsec $\rightarrow$ 2.5 Msymbols/sec $\rightarrow$ $n = 10$
  - $\odot$ $T = 4$ μsec $\rightarrow$ 0.25 Msymbols/sec $\rightarrow$ $n = 1$

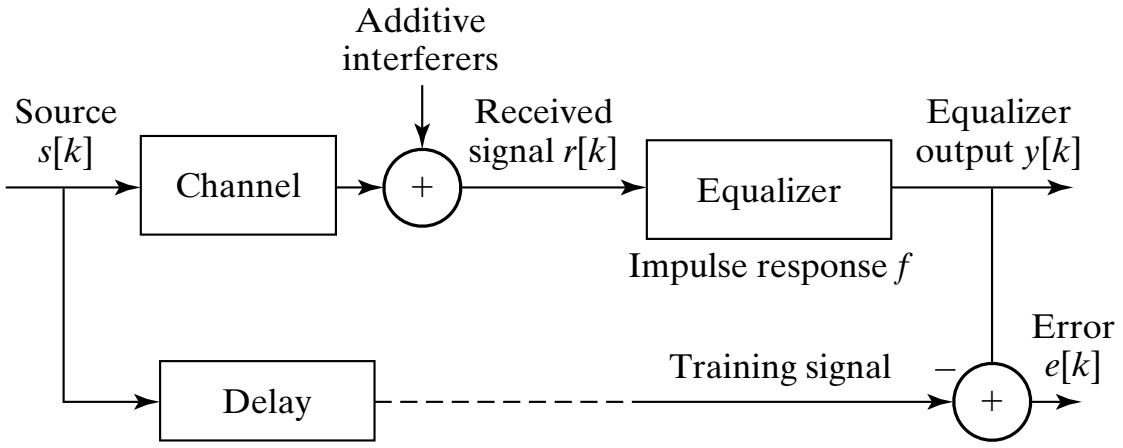
Multipath ... Interference (cont'd)

- Multipath FIR model coefficients depend on actual baud-timing choice of clock recovery algorithm, which need not match timing in non-ISI situation.
- Example: Two-ray analog channel $c(t) = p(t) + p(t - \Delta)$ with $\Delta = 0.7T$

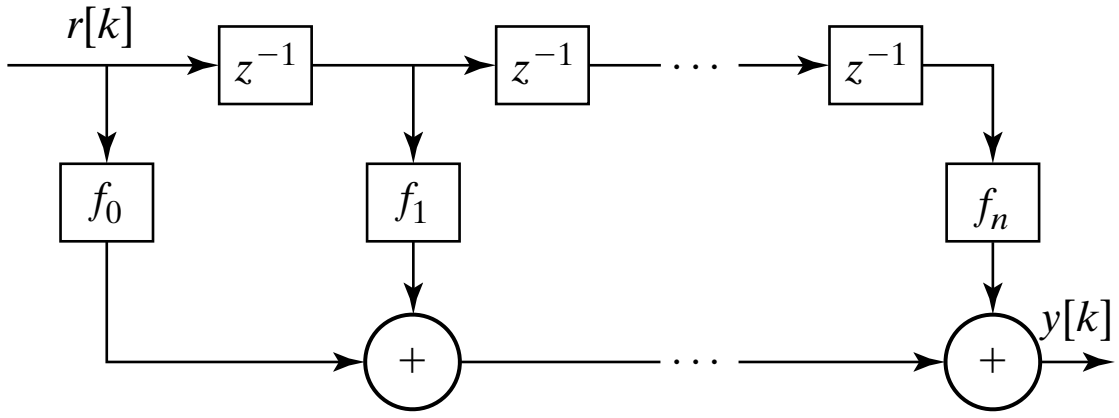

Lattice of T_s-spaced optimal sampling times with ISI
Lattice of T_s-spaced optimal sampling times with no ISI
Sum of received pulses
$c(t) = p(t) + 0.6\ p(t - \Delta)$
$p(t)$
$0.6\ p(t - \Delta)$
0
The digital channel model is given by T_s-spaced samples of $c(t)$

Trained Linear Equalization

- *Objective*: Given prearranged (intermittently transmitted) training sequence available at receiver, choose impulse response f of equalizer so $x[k] \approx s[k-\delta]$ (so $e \approx 0$) for some δ.



- *Equalizer Output*: $x[k] = \sum_{j=0}^{n} f_j r[k-j]$



Trained Adaptive Least-Mean-Square (LMS) Equalization

We choose to minimize

$$\text{avg}\{e^2[k]\} = \frac{1}{N}\sum_{k=k_0}^{k_0+N-1} e^2[k]$$

with $e[k] = s[k-\delta] - \sum_{i=0}^{n} f_i r[k-i]$ using a gradient descent scheme

$$f_i[k+1] = f_i[k] - \bar{\mu}\frac{\partial(\text{avg}\{e^2[k]\})}{\partial f_i}|_{f=f[k]}$$

With differentiation and average approximately commutable

$$f_i[k+1] \approx f_i[k] - \bar{\mu}\cdot\text{avg}\left\{\frac{\partial e^2[k]}{\partial f_i}|_{f=f[k]}\right\}$$

Dropping the "outer" average produces LMS

$$f_i[k+1] = f_i[k] - 2\bar{\mu}\left(e[k]\frac{\partial e[k]}{\partial f_i}\right)|_{f=f[k]}$$

$$= f_i[k] + \mu(s[k-\delta] - y[k])r[k-i]$$

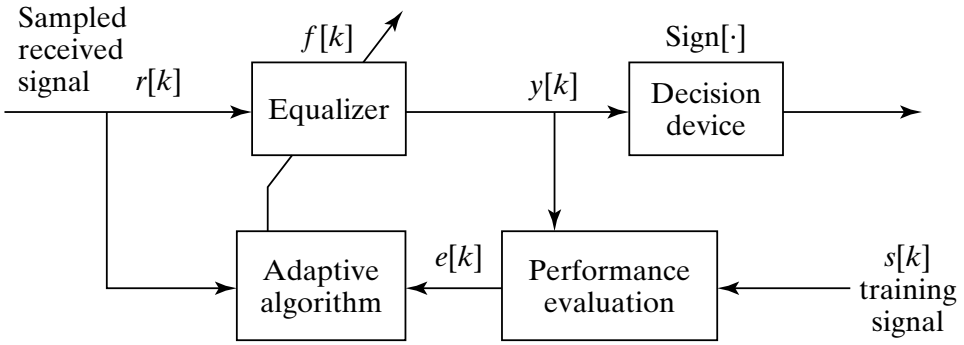
with $y[k] = \sum_{j=0}^{n} f_j[k]r[k-j]$.

Trained Adaptive Least-Mean-Square (LMS) Equalization (cont'd)

With the definition of the FIR equalizer output

$$y[k] = \sum_{j=0}^{n} f_j[k] r[k-j]$$

in



the trained approximate gradient descent adaptation algorithm LMS for the linear equalizer is

$$f_i[k+1] = f_i[k] + \mu(s[k-\delta] - y[k]) r[k-i]$$

- Should be engaged only during processing of portion of received signal due to training segment, e.g., using marker correlation.

Blind Adaptive Decision-Directed Equalization

We choose to minimize

$$\text{avg}\{(\text{Q}(\sum_{j=0}^{n} f_j r[k-j]) - \sum_{j=0}^{n} f_j r[k-j])^2\}$$

$$= \frac{1}{N} \sum_{k=k_0}^{k_0+N-1} (\text{Q}(\sum_{j=0}^{n} f_j r[k-j]) - \sum_{j=0}^{n} f_j r[k-j])^2$$

using a gradient descent scheme

$$f_i[k+1] = f_i[k] - \bar{\mu} \frac{\partial}{\partial f_i} \left(\text{avg}\{(\text{Q}(\sum_{j=0}^{n} f_j r[k-j]) \right.$$

$$\left. - \sum_{j=0}^{n} f_j r[k-j])^2\} \right) |_{f=f[k]}$$

Blind Adaptive Decision-Directed Equalization (cont'd)

Commute average and partial derivative, drop "outer" average, and presume $\partial(\mathrm{Q}(\sum_{j=0}^{n} f_j r[k-j]))/\partial f_i = 0$ to produce

$$f_i[k+1] = f_i[k] - 2\bar{\mu}\{(\mathrm{Q}(\sum_{j=0}^{n} f_j r[k-j])$$

$$-\sum_{j=0}^{n} f_j r[k-j])\frac{\partial(-\sum_{j=0}^{n} f_j r[k-j])}{\partial f_i}\}|_{f=f[k]}$$

$$= f_i[k] - 2\bar{\mu}\left(\mathrm{Q}(\sum_{j=0}^{n} f_j[k] r[k-j])\right.$$

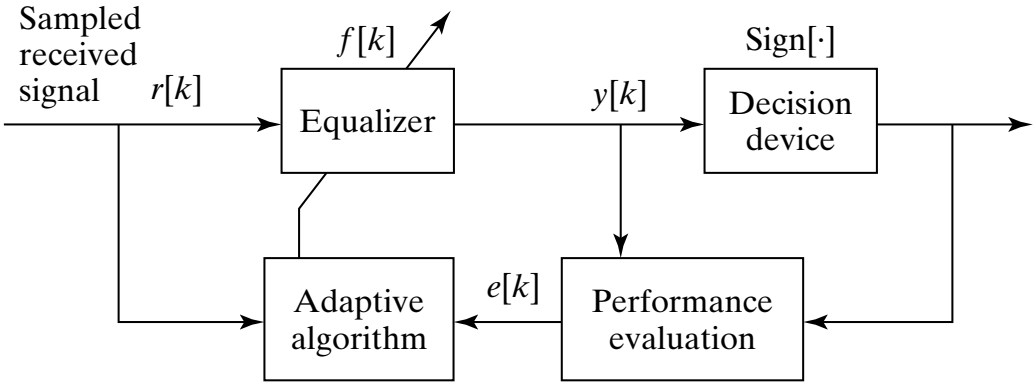
$$\left.-\sum_{j=0}^{n} f_j[k] r[k-j]\right)(-r[k-i])$$

Blind ... Equalization (cont'd)

With the definition of

$$y[k] = \sum_{j=0}^{n} f_j[k] r[k-j]$$

in



the decision-directed approximate gradient descent adaptation algorithm for the linear FIR equalizer is

$$f_i[k] = f_i[k] + \mu(\mathrm{Q}(y[k]) - y[k]) r[k-i]$$

- Relative to trained adaptation via LMS, the decision device output just replaces the training signal.

Blind Adaptive Dispersion-Minimizing Equalization

We choose to minimize

$$\text{avg}\{(1-(\sum_{j=0}^{n} f_j r[k-j])^2)^2\}$$

$$= \frac{1}{N}\sum_{k=k_0}^{k_0+N-1}(1-(\sum_{j=0}^{n} f_j r[k-j])^2)^2$$

using a gradient descent scheme

$$f_i[k+1] = f_i[k]$$

$$-\bar{\mu}\frac{\partial\left(\text{avg}\{(1-(\sum_{j=0}^{n} f_j r[k-j])^2)^2\}\right)}{\partial f_i}|_{f=f[k]}$$

Commuting average and differentiation and dropping "outer" average produces

$$f_i[k+1] = f_i[k] + 2\bar{\mu}\{(1-(\sum_{j=0}^{n} f_j r[k-j])^2)$$

$$\cdot\frac{\partial(\sum_{j=0}^{n} f_j r[k-j])^2}{\partial f_i}\}|_{f=f[k]}$$

Blind ... Equalization (cont'd)

Evaluating derivative produces

$$f_i[k+1] = f_i[k] + \mu(1 - (\sum_{j=0}^{n} f_j[k]r[k-j])^2)$$

$$\cdot(\sum_{j=0}^{n} f_j[k]r[k-j])r[k-i]$$

where

$$\sum_{j=0}^{n} f_j[k]r[k-j] = y[k]$$

so

$$f_i[k+1] = f_i[k] + \mu(1 - y^2[k])y[k]r[k-i]$$

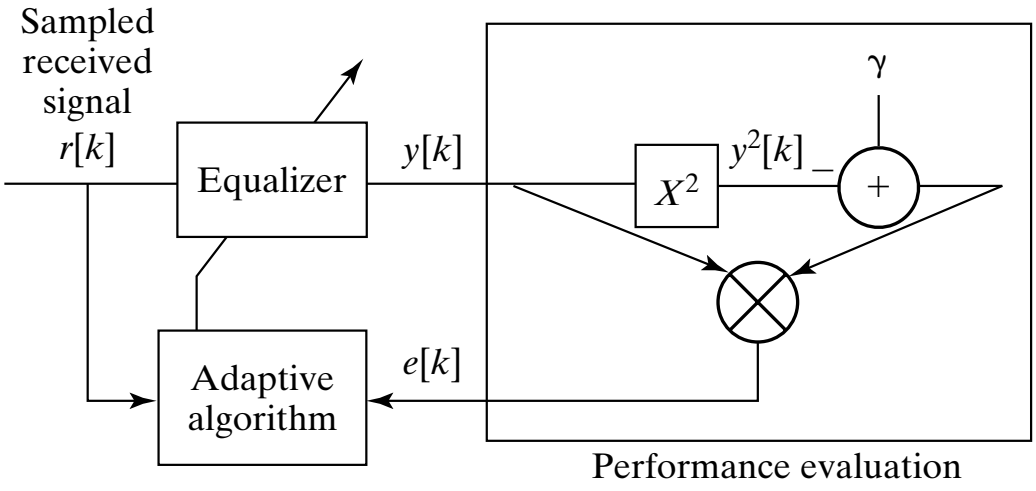
In comparison to LMS the prediction error $s[k-\delta] - y[k]$ has been effectively replaced by $(1 - y^2[k])y[k]$.

Blind ... Equalization (cont'd)

With the definition of

$$y[k] = \sum_{j=0}^{n} f_j[k] r[k-j]$$

in



the dispersion-minimizing approximate gradient descent adaptation algorithm for the linear FIR equalizer is

$$f_i[k+1] = f_i[k] + \mu(1 - y^2[k]) y[k] r[k-i]$$

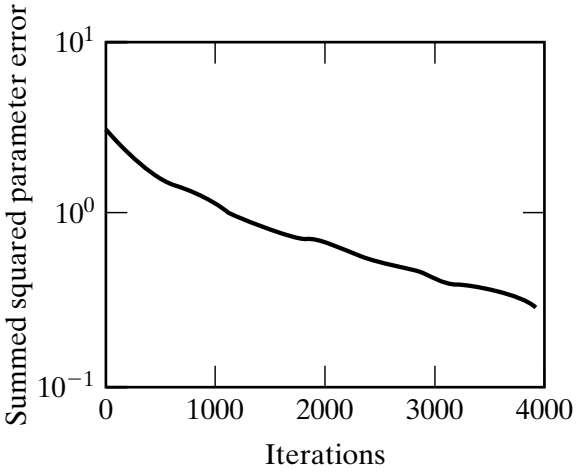
- The adaptive scheme is labelled as blind (rather than trained) due to the creation of the correction term without a training signal.

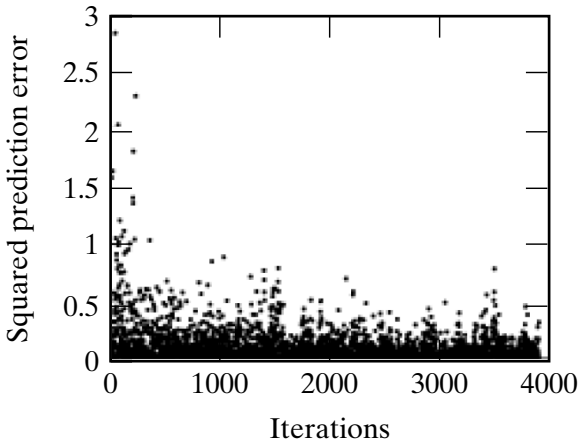
Example (using `dae`)

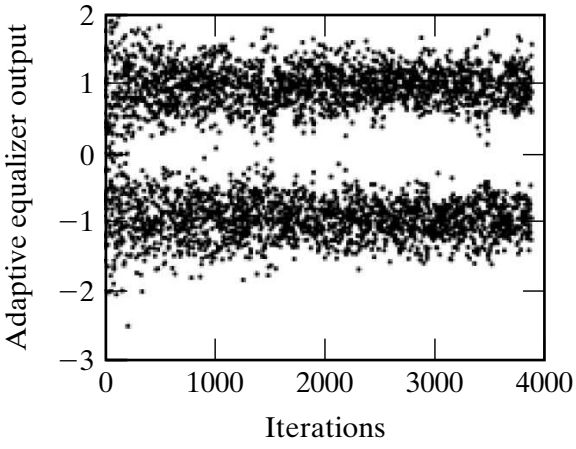
- Source: binary (± 1)
- Channel Impulse Response: {1 .9 .81 .73 .64 .55 .46 .37 .28}/4.138
- Sinusoidal interferer frequency: 1.4 radians/sample
- Some broadband noise present
- Equalizer length: 33

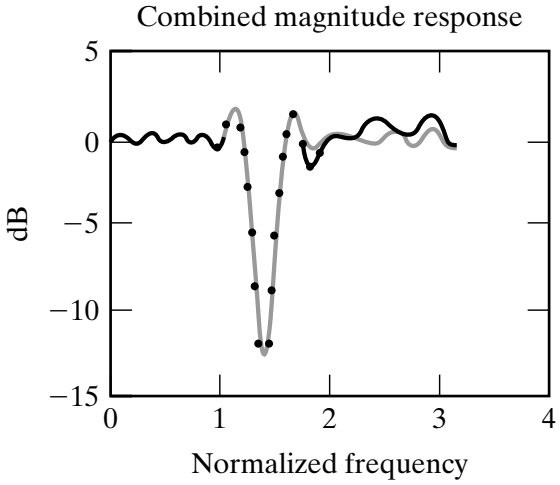
Example (cont'd)

Trained LMS:



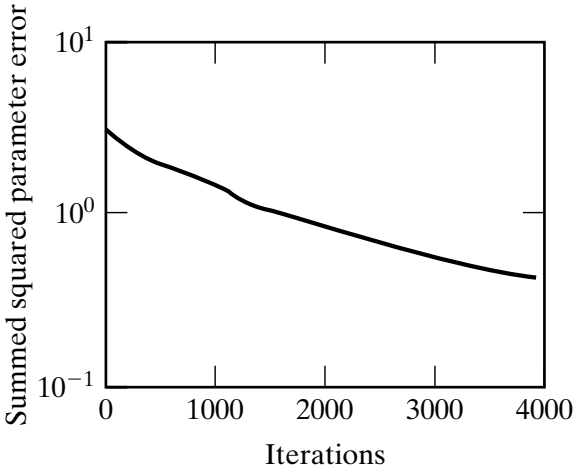


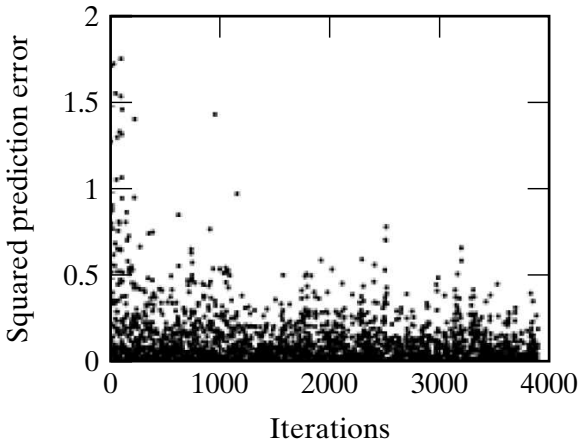


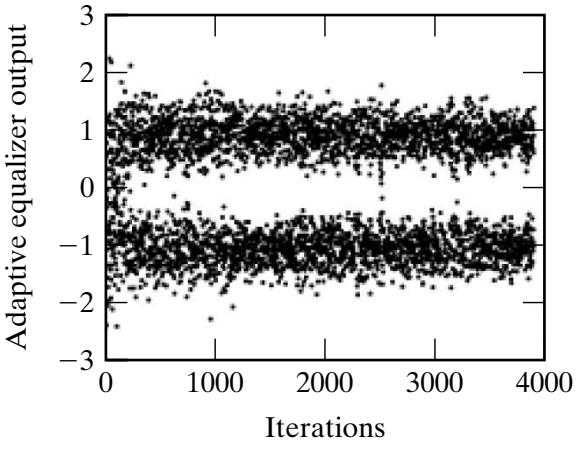


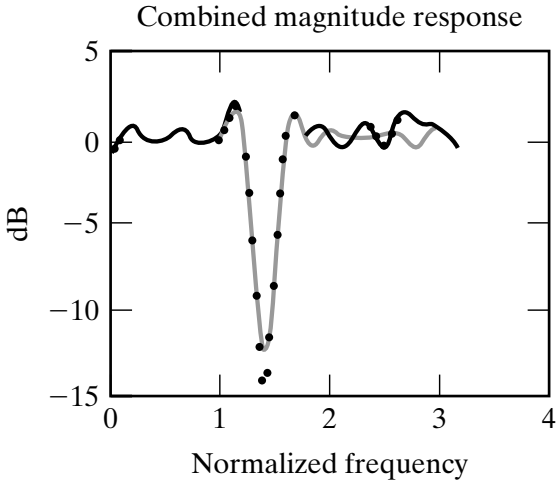
Example (cont'd)

Decision-directed:



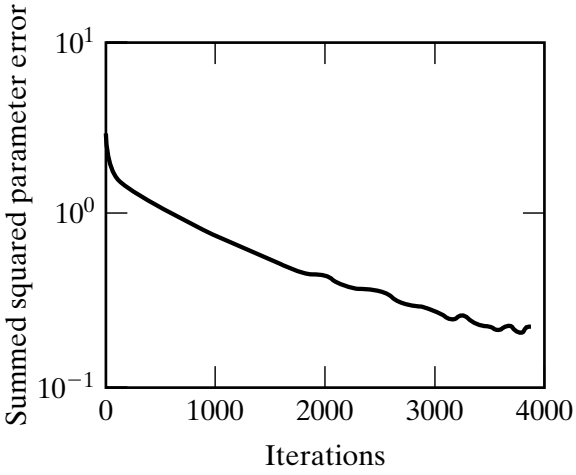


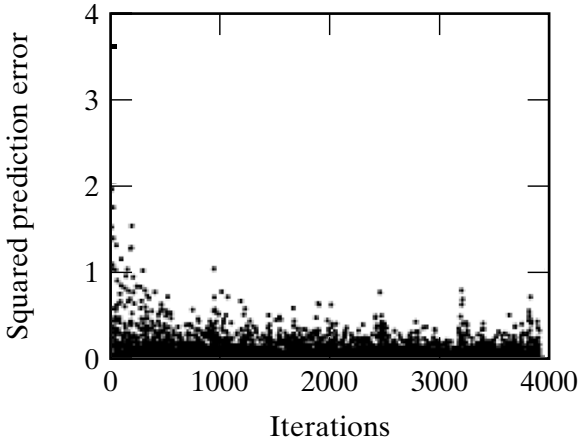


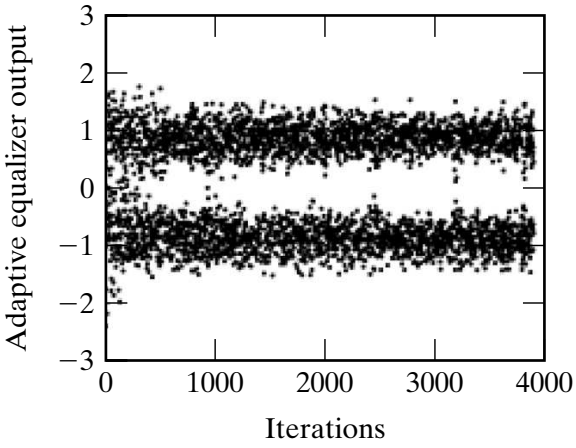


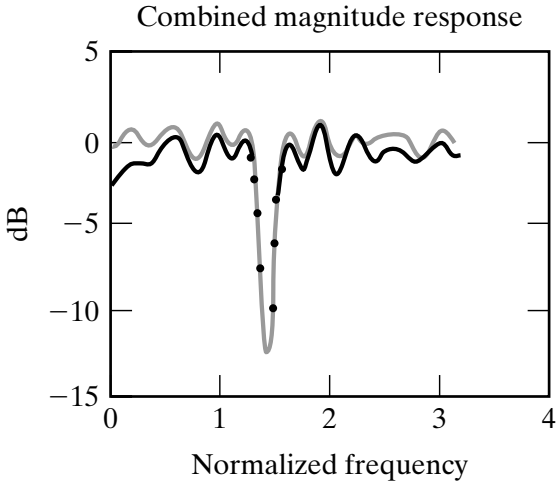
Example (cont'd)

Dispersion minimization:









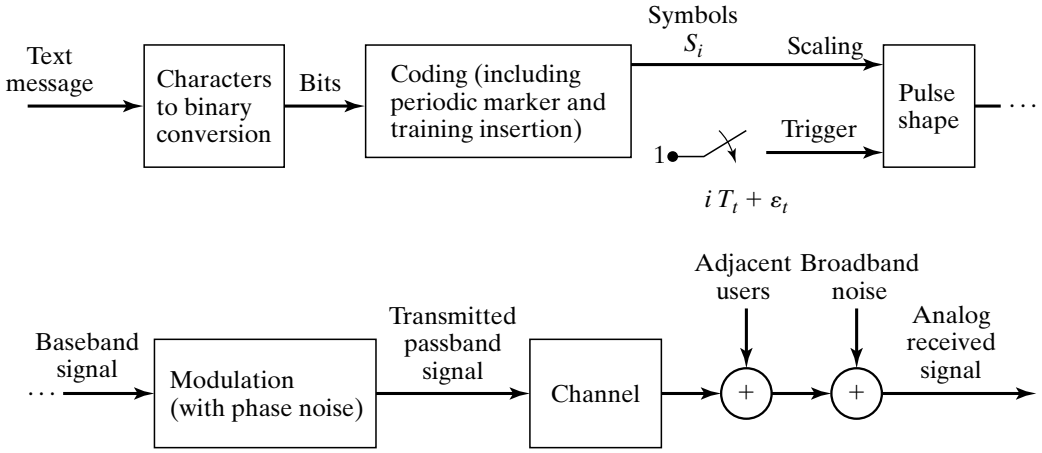
PUTTING IT ALL TOGETHER: RECEIVER DESIGN

- ⋆ Received Signal Construction
- ⋆ Receiver Design Methodolgy (in 4 Stages)

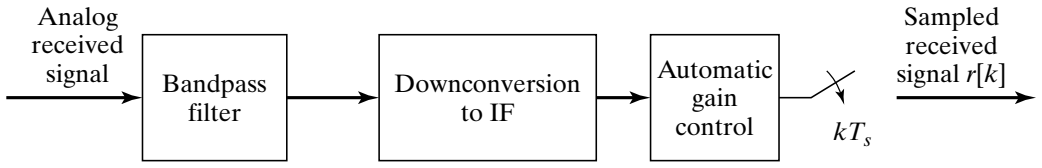
Received Signal Construction

Receiver design responds to the received signal composition.

Transmitter and channel:



Receiver front-end:



Received Signal Construction (cont'd)

- Original character string message is coded into 7-bit ASCII format and mapped to 4-PAM.
- Symbol sequence is composed as a 124-symbol marker/training segment, followed by 400 4-PAM message symbols, followed by the same 124-symbol marker/training segment, followed by another 400 message symbols, etc.
- Transmitter pulse period T_t precisely matches the symbol period specification adopted by receiver.
- Transmitter pulse-firing trigger (or baud-timing) offset ϵ_t is unknown to receiver.
- Pulse shape is truncated SRRC with rolloff factor of 0.3.

Received Signal Construction (cont'd)

- Frequency division multiplexing slots exceed double half-power bandwidth of pulse shape.
- Transmitter carrier frequency is known precisely at receiver.
- Transmitter carrier phase unknown to receiver and expected to be slowly wandering
- The channel can possess eye-closing ISI.
- Only the maximum delay spread of the potential ISI is known to the receiver in advance.
- Broadband noise is present, but modest.
- Narrowband interferers may be present as well.

Received Signal Construction (cont'd)

- Downconversion to IF by front-end hits specified target frequency exactly.
- Automatic gain control in front end is presumed converged and static.
- Sampler is free-running at a frequency well over twice the bandwidth of the pulse shape.
- Sampler is sub-Nyquist for IF, which means that downconversion will be performed on passband spectrum replica nearest baseband.

Received Signal Construction (cont'd)

Received Sampled Signal Specifications Table (left column):

symbol source alphabet
assigned intermediate frequency
nominal symbol period
SRRC pulse shape rolloff factor
FDM user slot allotment
truncated width of SRRC pulse shape
frame marker/training sequence
frame marker sequence recurrence period
time-varying IF carrier phase
IF frequency offset
transmitter baud timing offset
transmitter symbol period offset
channel delay spread maximum
sampler frequency

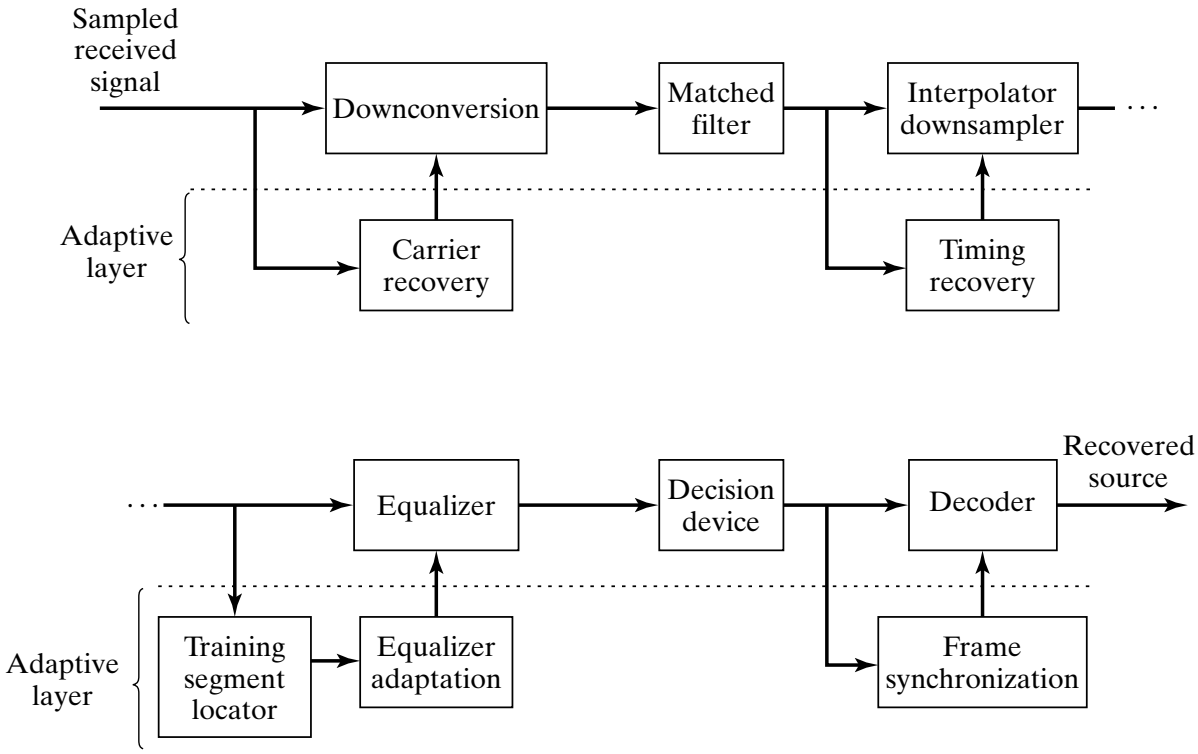
Received Signal Construction (cont'd)

Received Sampled Signal Specifications Table (right column):

±1, ±3
2 MHz
6.4 microseconds
0.3
204 kHz
8 transmitter clock periods
Ï0àâéOh well whatever Nevermind
524 symbols
lowpass filtered white noise
none
fixed
none
7 symbols
850 kHz

Receiver Design Methodology

- Stage One: Ordering the basic operations
- Stage Two: Selecting components
- Stage Three: Countering anticipated impairments
- Stage Four: Tuning and testing



Design Methodology (cont'd)

Stage One: Ordering the basic components

- The basic receiver components are
  - ⊙ downconversion with carrier recovery
  - ⊙ baud-timing recovery with matched filter and interpolator/downsampler
  - ⊙ trained equalizer with training segment locator
  - ⊙ decision device and decoder with frame synchronization
- Our ordering (downconversion, timed downsampling, equalization, and decoding) is classical and popular but not the only possibility.

Design Methodology (cont'd)

Stage One: Ordering the basic components (cont'd)

- Timing and equalization can occur in the passband before carrier recovery.
- A fractionally-spaced equalizer can absorb the matched filter and resampling operations of the baud-timing component.
- Sometimes ordering is based on design tradeoffs at hand, sometimes on designer preference or personal experience, and sometime's on factors outside receiver designer's control (e.g. legacy product lines and intellectual property constraints).

Design Methodology (cont'd)

Stage Two: Selecting components

- Downconversion (like the other operations of basic components) can be done through many methods.
- Here the sub-Nyquist sampling of the IF signal places replicas closer to baseband.
- The closest is to be downconverted by a mixer (with an adapted phase) followed by a suitable lowpass filter.
- The presumption is that the components chosen, when properly tuned, result in acceptable performance.
- The proper operation of the components selected can be confirmed by simulations in an interference-free, ideal/full-knowledge setting.

Design Methodology (cont'd)

Stage Three: Countering anticipated impairments

- ◦ residual interference from adjacent FDM band signals
- ◦ AGC jitter
- ◦ quantization noise in sampler
- ◦ round-off noise in filters
- ◦ residual interference from doubly upconverted spectrum
- ⊕ carrier phase jitter
- ⊕ baud timing offset
- ⊕ residual MSE from equalizer
- ⊕ equalizer parameter jitter
- ⊕ noise enhancement by equalizer

[*Legend*: ⊕ major; ◦ minor]

Design Methodology (cont'd)

Stage Three: Countering ... impairments (cont'd)

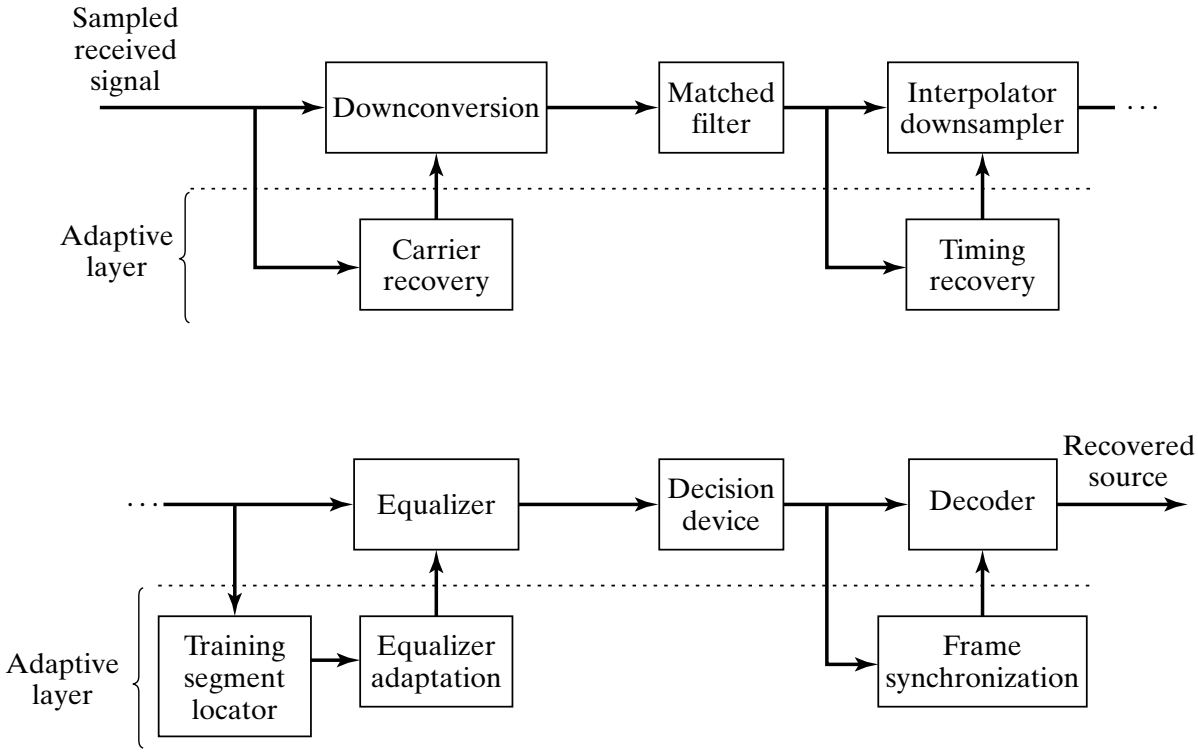
- We anticipate the need for
  - ⊙ carrier phase adaptation
  - ⊙ baud-timing adaptation
  - ⊙ equalizer adaptation
  - ⊙ post-decision frame synchronization
- Choices (so far)
  - ⊙ Carrier phase recovery: phase-locked loop and Costas loop
  - ⊙ Baud-timing recovery on oversampled matched filter output: output power, absolute value, fourth power, and dispersion
  - ⊙ Equalizer adaptation: trained LMS, decision-directed, dispersion-minimizing
  - ⊙ Frame synchronization (and training segment location): marker correlation

Design Methodology (cont'd)

Stage Four: Tuning and Testing

In order of appearance:

- Step One: Tuning the Carrier Recovery
- Step Two: Tuning the Clock Recovery
- Step Three: Tuning the Equalizer
- Step Four: Frame synchronization for decoder



Design Methodology (cont'd)

Stage Four: Tuning and Testing (cont'd)

Plan of action:

- One at a time
- In order of appearance
- With preceding steps countering their impairments as intended
- Each with its own share of total allowable error

Design Methodology (cont'd)

Stage Four: Tuning and Testing (cont'd)

Tuning tradeoffs:

- All adaptive components will select stepsize in tradeoff between rapid tracking and dampened jitter.
- Carrier recovery
  - ⊙ LPF cutoff frequency and range between in-band and stopband gain
- Clock recovery
  - ⊙ δ in derivative
  - ⊙ time support of interpolation filter

Design Methodology (cont'd)

Stage Four: Tuning and Testing (cont'd)

- Equalizer
  - ⊙ number of taps: channel inverse delay spread; 2 to 5 times channel maximum delay spread
  - ⊙ training signal delay: half of equalizer length
  - ⊙ initialization: center spike
- Frame (or training) synchronization
  - ⊙ marker chosen for peaky autocorrelation
  - ⊙ preferred marker unlikely to occur in message

Development Tips

- simulate transmitter to allow controlled tests on broader set of circumstances than provided by test signal set
- probe receiver limits (e.g. assess how much noise causes performance failure)
- implement debug mode that plots pertinent signals
- test an adaptive element in two scenarios:
  - (i) start at right answer with zero stepsize and see if achieved performance is as expected, and then
  - (ii) start near right answer with nonzero stepsize and see if algorithm shrinks into tight orbit about right answer

Laboratory Exercises – Day 4

Introduction to Digital Communication Receiver Design

Implementation of Decision-Directed LMS

The existing receiver only uses the trained LMS algorithm for equalizer adaptation. Your task is to add the decision-directed (DD) LMS algorithm to the receiver. Note that you should *not* remove the trained LMS algorithm which operates during `TRAINING_MODE`. Rather, your task is to *add* the DD-LMS algorithm which will operate when in `DATA_MODE`.

As before, you should start with the most benign channel conditions, then gradually increase the impairment. After successfully adding the DD algorithm, you should tune the stepsize. Then, complete the following tasks:

1. Static channel

   - Starting with the default simulation parameters, set the trained LMS stepsize and DD-LMS stepsizes to zero, which effectively disables the equalizer adaptation. Change the channel in `main.m` to `c=[1, -0.6, 0.3]`. Run `main.m`, and plot the smoothed, squared DD equalizer output error on the same plot as the smoothed squared LMS error (but in a different color). Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?
   - Keep the DD-LMS stepsize at zero, but set the *trained* LMS stepsize to 0.001. Re-run `main.m`. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?
   - Now set both the DD-LMS stepsize and trained LMS stepsizes to 0.001. Re-run `main.m`. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?
   - Finally, set the DD-LMS stepsize to 0.001, but the trained LMS stepsize to zero. Re-run `main.m`. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?

2. Time-varying Channel

   - Load the file `/day4/time_var.mat` which contains a signal from a time-varying channel, stored in the variable `r`. Set the trained LMS stepsize and DD-LMS stepsizes to zero. Test your receiver on the signal. Is your receiver able to track the time-varying channel? Show a plot of the equalizer output error, during both trained and DD modes. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?
   - Keep the DD-LMS stepsize at zero, but set the *trained* LMS stepsize to 0.001. Test your receiver on the signal. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?

- Keep the trained LMS stepsize set to 0.001, and pick your own stepsize for the DD-LMS algorithm. Test your receiver on the signal, and tune the stepsize to your liking. Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? Is the equalizer able to track? What is the BER?
- Finally, set the stepsize of the trained LMS algorithm to zero. Test your receiver on the time-varying signal again. Is it able to track now? Is the eye open by the end of the simulation? If so, at approximately which iteration is the eye open? What is the BER?

Final Project - Day 5

Introduction to Digital Communication Receiver Design

Description of the project

- The transmitter/receiver code we have been using (in the `/system_code` directory) was developed by a former employee at the company where you work. As such, development of the radio has not progressed for some time. Meanwhile, your manager has just informed you that the chief competitor to your company has released a radio with superior performance. Your manager has made several suggestions in hopes of improving upon the existing sampled-IF receiver. You will use code (and knowledge!) that you have developed in the lab to modify the existing MATLAB simulation to build a receiver that performs better than the competition.

- The original radio was based on a draft ETSI standard that had not yet been ratified. A standards battle ensued, and when the standard was finally ratified, some of the parameters of the radio changed. While the radio is still a 4-PAM radio (thanks to the strength of your marketing department), these are the parameters that have changed:

parameter	value
assigned intermediate frequency	2.2 MHz
nominal symbol period	5 microseconds
frame marker	ÑÝÙÇKñÿúçk
training sequence	LarryCurlyMoe
training sequence recurrence period	472
SRRC pulse shape rolloff factor	0.25
sampler frequency	1 MHz

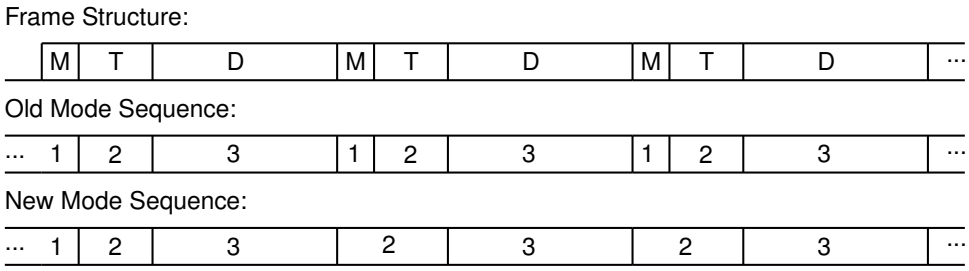
  These parameters can be found in the file `/day5/globalParams_project.m`.

- To improve the performance of your receiver, your manager asks you to make the following changes to `Rx.m` (which were already completed in the lab):

  1. Change the carrier recovery scheme. It is currently implemented using a PLL, and you are asked to change it to a Costas loop (refer to lab from Day 2).
  2. Add a decision-directed equalizer. Currently, the equalizer is only adapted during the training periods. Your manager believes your company will have a competitive advantage if you are able to make the radio work well in environments where the channel is time-varying (refer to lab from Day 4).

- Sending large amounts of training data reduces the effective throughput of the radio, so the new standard has shortened the length of the training sequence. This comes at a cost, however, as the equalizer may not have enough data to "open the eye".

Fortunately, your manager has another wise suggestion, and has drawn a diagram which shows the frame structure of the transmitted signal, consisting of the marker (M), training (T), and data (D). Recall from the documentation of the MATLAB code that the receiver operates in 3 modes: (1) header search mode, (2) training mode, and (3) data mode. These modes are also shown in the figure.



Your manager has pointed out the following fact: after the first marker sequence has been found, there is no reason to continue looking for subsequent marker sequences since you know the length of each frame, and therefore you know the location of the next training sequence. Thus, your manager requests that you make the following changes:

  - After you find the first marker sequence, turn off the correlator. This way, the receiver will burn less power since the correlator only needs to operate at the start of the reception.
  - Since you know the marker sequence, you can use it to help train the equalizer (thereby compensating for the reduced amount of training data). Instead of operating in the old mode sequence 1,2,3,1,2,3,..., modify the receiver so that it uses the new mode sequence 1,2,3,2,3,....

- In the `/day5` directory, you will find 3 test vectors: `easy.mat`, `medium.mat`, and `hard.mat`. Each of these MATLAB data files contains an example received signal, and each originates from an increasingly hostile communication environment. Using the file `tester.m`, you can test the performance of your receiver. You should place all files in the same directory.

Evaluation

- At the end of class, you will be presented with a mystery signal. You will be assigned a grade based on how many errors your receiver makes. You are not required to decode the message contents for the first 5 frames. This is to allow your algorithms time to converge. All symbols after the first 5 frame will be used to calculate your grade. The "difficulty" of the mystery signal will be between that of the `medium.m` and `hard.m`.

- Your grade will be based on the output of the `tester.m` program, and it is your responsibility to make sure that your receiver is compatible with this script. If your

receiver does not operate with any of the provided test vectors, then it certainly will not work with the mystery signal.

- You will be required to explain to the instructor the operation of your MATLAB code (in `Rx.m` only).
- Your are also required to include a very brief report (1-2 pages) detailing the performance of your receiver on the 3 test vectors (`easy.m`, `medium.m`, and `hard.m`). In this report, you should also include the following plots for `medium.m` only:
  - Carrier phase
  - Timing offset
  - Equalizer error

  If your receiver makes errors on any of the test vectors, you should make a conjecture about *why* your receiver is unable to make zero errors. Is there a particular component of the receiver that seems to be the source of the errors?
- The receiver that you design must be your own work.

Suggestions

- You may find it easiest to add the requested modifications incrementally. Testing your code after each change will help narrow down the possible sources of an error.
- You may have to do some adjustment of algorithm stepsizes in your receiver. This is a natural part of the design process.
- Start with the `easy.mat` test vector. Once your simulation works with this vector, you should progress to the `medium.mat` test vector, and then to `hard.mat`.
- Try to break your receiver. See how much noise can be present in the received signal before accurate demodulation seems impossible (e.g. BER $> 10^{-2}$). Try to determine how bad the worst channel can be through which a signal can be transmitted where your receiver correctly decodes the signal.
- The test vectors (and mystery signal) may have originated from a time-varying channel. Take note of the ability of your equalizer to track by looking at the equalizer error signal. Does the error stay small? Or does it increase? Again, you will want to tune the stepsize.